Ingeniería Técnica de Telecomunicación
Especialidad en Telemática

PROYECTO FIN DE CARRERA

HealthCoach: Desarrollo de una aplicación móvil dentro del campo del m-health

Autor: David López Muñoz

Tutor: Dr. David Griol Barres

Leganés, octubre de 2015

<La imaginación es más importante que el conocimiento.>

Albert Einstein

Agradecimientos

La entrega de este proyecto supone cerrar una importante etapa de mi vida, por ello quiero agradecer el apoyo recibido a lo largo de la realización del mismo, y de forma general en mi etapa como estudiante de la Universidad.

Quiero expresar mi agradecimiento por la ayuda, el criterio, y el constante ánimo proporcionados por mi tutor, David Griol.

Especial mención a toda mi familia, por su cariño, y por recordarme que era menester finalizar el proyecto. Quiero transmitir mi gratitud a mis padres, Pascual y Seve, por brindarme la oportunidad de iniciar mis estudios e inculcarme lo valioso que es formarse académicamente, y a mis hermanos, César y Almudena, por ser un referente en mi vida.

Por último quiero dar las gracias a mi pareja, Amara, por su confianza, apoyo incondicional y palabras de aliento en los buenos y malos momentos.

Resumen

En este Proyecto Fin de Carrera se ha desarrollado un sistema estándar dentro del campo del m-health, destinado al tratamiento de diferentes dolencias médicas mediante el uso de dispositivos móviles de la plataforma Android. Este sistema, denominado HealthCoach, implementa una aplicación que presenta al paciente una serie de tarjetas a modo de ejercicios terapéuticos que permiten una interacción multimodal mediante el uso del reconocimiento de voz y la síntesis de texto a voz, además del uso táctil de la aplicación. La segunda aplicación, tiene el propósito de monitorizar la evolución del paciente en el proceso terapéutico, por lo tanto va dirigida a los profesionales sanitarios o familiares del enfermo.

En esta memoria se ha elegido realizar un ejemplo práctico sobre la enfermedad de Alzheimer, diseñando e implementando tarjetas que preserven las capacidades cognitivas de las personas que sufren esta dolencia. Estas tarjetas son sometidas a un proceso de adaptabilidad en el grado de dificultad y tipo de ámbito cognitivo presentados al paciente, basado en los resultados obtenidos en el transcurso de la terapia. Al finalizarla, la aplicación estimará de forma automática el grado de enfermedad del usuario. Para ello, se ha realizado un estudio de las técnicas de terapia y diagnóstico asociadas al Alzheimer.

Además, se justifica la elección de Android frente a otras plataformas, realizando un estudio de su arquitectura y detallando las posibilidades que ofrece para implantar un sistema multimodal. Se pretende que este proyecto sirva de referencia para el desarrollo de otras aplicaciones Android, detallando todo el ciclo de desarrollo del software.

El sistema desarrollado cuenta con una arquitectura cliente-servidor, nutriéndose de varias tecnologías para persistir la información en el servidor remoto, alojado en un dispositivo Raspberry Pi. La solución propuesta utiliza una API REST desarrollada en PHP y alojada en un servidor de aplicaciones Apache, para facilitar la interacción con la base de datos MySQL.

Palabras clave: Android, *m-health*, Alzheimer, Interacción multimodal, PHP, MySQL, Apache, *Raspberry Pi*, REST, API, JSON.

Abstract

In this Final Degree Project, a standard M-Health system has been developed. It is intended to be used in the treatment of different medical diseases, using Android mobile devices. This system, named HealthCoach, uses an app which shows to the patient a number of cards displaying therapeutic exercises, allowing multi-modal interaction by using voice recognition and speech synthesis as well as tactile interaction. A second app is used to monitor the evolution of patients throughout their therapeutic treatment, hence it is intended to be used by healthcare professionals and patients' relatives.

This statement exposes a practical example based on Alzheimer's disease, in which a number of cards has been designed and built aiming to preserve the cognitive capabilities of the patients. These cards are modulated by level of complexity and cognitive domain, according to the evolution of the results of the therapy. When the therapy is complete, the app estimates the stage of the disease. In order to do that, therapy and diagnosis techniques related to Alzheimer’s disease have been studied.

It is also justified the election of Android instead of other platforms, examining its architecture and detailing the possibilities it offers to implement a multi-modal system. This project intends to serve as a reference for developing other similar Android apps, so it explains in detail the software development cycle used.

This new system has a client-server architecture, making use of a number a technologies to persist the information in the remote server, which is hosted in a Raspberry Pi device. The proposed solution uses a REST API implemented with PHP, running in an Apache application server that helps interacting with a MySQL database.

Keywords: Android, *m-health*, Alzheimer, multi-modal interaction, PHP, MySQL, Apache, *Raspberry Pi*, REST, API, JSON.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

Este capítulo dedica sus líneas a introducir de forma breve el presente Proyecto Fin de Carrera, exponiendo cuál es su motivación, qué objetivos persigue desde su concepción y cuál es la estructura que compone esta memoria. Después expone la planificación temporal, un listado de los recursos utilizados y el presupuesto final de este proyecto.

1.1 Motivación

La rápida proliferación de las comunicaciones móviles y en especial de los "**teléfonos inteligentes**" (del inglés *smartphone*) durante la última década, ha convertido a éstos en una realidad cotidiana para millones de personas, que no solo ha cambiado la forma de comunicarnos, sino que también ha abierto un abanico de nuevas posibilidades para informarnos, aprender, aumentar nuestra productividad o por qué no, de monitorizar, motivar y expandir nuestros hábitos e intereses. En España, el 87,8% de la población mayor de 15 años utiliza de forma habitual un *smartphone*, alcanzando un grado de implantación en los hogares del 95,3% [1].

Uno de los sectores que está evolucionando de forma notoria en los últimos años para beneficiarse de las TIC (Tecnologías de la Información y Comunicación) es el **sanitario**, evolucionando hacia un modelo digital que facilite la compartición de historiales clínicos, imágenes médicas y en general de cualquier tipo de información involucrada en el sector salud. Es el llamado ***e-health***, cuya implantación en España es una de las más altas a nivel europeo, consiguiendo por ejemplo que las solicitudes de citas médicas realizadas a través de Internet supongan un 54% del total [2]. Dentro del *e-health*, el servicio con un mayor impacto actualmente es el ***m-health*** o servicios sanitarios a través de dispositivos móviles, que a día de hoy son utilizados por un 51% de los médicos y que en 2015 serán utilizados por 500 millones de usuarios a nivel mundial.

A pesar de la gran aceptación de *smartphones* y de servicios como el *e-health* en la sociedad, las personas mayores sufren una **brecha digital** en relación al acceso y uso de las nuevas tecnologías, cuyo grado de exclusión no es alcanzando por ningún otro grupo de edades.

1.2 Objetivos

Este Proyecto Fin de Carrera tiene como principal objetivo crear una aplicación estándar ubicada en el marco del ***m-health***, que pueda adaptarse fácilmente al tratamiento de diversas **dolencias médicas**. Como ejemplo práctico para desarrollar en esta memoria, se utilizan una serie de ejercicios terapéuticos centrados en las funciones cognitivas de los enfermos de **Alzheimer**, con el propósito de mantener y potenciar esas capacidades. A su vez, se pretende facilitar a los terapeutas y/o familiares la monitorización y diagnóstico de los pacientes que realizan la terapia cognitiva.

Otro objetivo, no menos importante, es el de romper la brecha digital existente en las personas mayores, definiendo una aplicación usable y con mecanismos de interacción lo más naturales posibles. Por ello, se define el requisito de utilizar un sistema de **reconocimiento de voz** y **síntesis de texto a voz** en el sistema a desarrollar, que complemente a la interacción táctil para lograr un **sistema multimodal**.

En la fase inicial de este proyecto se ha recopilado información referente a los principales sistemas operativos móviles. El propósito es realizar una comparativa de funcionalidades y seleccionar el que mejor se adapte a los requerimientos de la aplicación que será implementada. La elección final recae en **Android** [3], por lo que otro de los objetivos consiste en estudiar su arquitectura y principales características, así como servir de ejemplo para el desarrollo de otras aplicaciones que se apoyen en un **servicio WEB** para almacenar y recuperar la información de una base de datos externa.

Adicionalmente, se pretende eliminar la dependencia de recurrir a un servidor externo que proporcione alojamiento para el sistema desarrollado, utilizando un dispositivo físico de bajo coste que realice las funciones de servidor de aplicaciones y bases de datos.

También se persigue el objetivo de realizar un proyecto con elementos diferenciadores con el resto de aplicaciones Android centradas en el Alzheimer, como son la **adaptabilidad** de los ejercicios presentados en función de los resultados del paciente y la realización de un **cálculo estimado** del grado de la enfermedad

Como fin último, se encuentra el de abrir el camino hacia nuevas funcionalidades o desarrollos basados en el proyecto implementado, tal y como se describe en el **Capítulo 8** de esta memoria.

1.3 Estructura de la memoria

En referencia a la estructura del presente documento, la memoria se compone de ocho capítulos, dos anexos, glosario de términos y bibliografía. A continuación se resume el contenido de cada uno de ellos:

- **Capítulo 1. Introducción:** El primer capítulo incluye la motivación del proyecto, los objetivos marcados y la estructura general de la memoria. La gestión del proyecto se detalla en este capítulo mediante la planificación temporal de las diferentes tareas que componen el proyecto, un listado de los recursos hardware y software, finalizando con un desglose de los costes personales y materiales empleados para generar la cifra total del coste del proyecto.

- **Capítulo 2. Estado del arte:** El capítulo estudia de forma más detallada la evolución de los dispositivos móviles, el amplio abanico de sistemas operativos que los gobiernan y la justificación de la elección de Android como plataforma para desarrollar la aplicación. Continúa con un estudio de los sistemas de diálogo y analiza, a modo de ejemplo, aplicaciones que los utilizan. Después enumera las posibilidades que brinda la plataforma Android en el reconocimiento de voz y síntesis de texto a voz, con un estudio comparativo de los principales motores de síntesis existentes en el mercado. Finaliza con el estudio de las técnicas de terapia y diagnóstico aplicadas a la enfermedad de Alzheimer, complementado con un análisis de las principales aplicaciones Android destinadas a esta dolencia.

- **Capítulo 3. Tecnologías y herramientas:** Este capítulo presenta una breve descripción de las principales tecnologías y herramientas software utilizadas en el despliegue y desarrollo del proyecto.

- **Capítulo 4. Sistema operativo Android:** Este capítulo se dedica en exclusiva a analizar la arquitectura y componentes del sistema operativo Android, cuya elección se justifica en el Capítulo 2.

- **Capítulo 5. Análisis y diseño:** El capítulo incluye la introducción al sistema HealthCoach, los requisitos de usuario y casos de uso propuestos, para continuar con el análisis de la arquitectura general del proyecto basada en dos aplicaciones cliente Android y un servidor de aplicaciones y base de datos, que gestiona la persistencia y acceso a los datos. A continuación, se muestra el diagrama de clases y se detalla el diseño de la interfaz de usuario compuesta por las plantillas de los diversos tipos de pantallas involucrados. Además se enumeran los distintos tipos de tarjetas de ejercicios de terapia, basados en una seria de ámbitos cognitivos seleccionados. Para finalizar se explica el diseño del algoritmo de adaptabilidad de la terapia y del cálculo de grado de la enfermedad.

- **Capítulo 6. Desarrollo**: Este capítulo describe la implementación de las funcionalidades generales y módulos específicos utilizados en el sistema diseñado en el Capítulo 5. Las funciones generales comprenden la ejecución asíncrona de tareas mediante AsyncTask, implementación e invocación de una API REST en lenguaje PHP, lectura de objetos JSON, uso de la librería JavaMail-Android para envío de correos electrónicos, implementación de reconocimiento y síntesis de voz, comprobación del estado de la conexión e interfaz de usuario. En cuando a los módulos, se detalla la implementación de la identificación y registro, menú principal, tarjetas cognitivas, módulo estadístico y de cálculo del estadio de Alzheimer.

- **Capítulo 7. Evaluación:** En este capítulo se describe el diseño de los cuestionarios utilizados para realizar la evaluación de las aplicaciones de pacientes y responsables, involucradas en la terapia y diagnóstico de Alzheimer. Se presentan además los resultados de la evaluación basada en los cuestionarios anteriores.

- **Capítulo 8. Conclusiones y futuras líneas de trabajo**: En el último capítulo se exponen las conclusiones finales obtenidas con la realización del proyecto, así como las líneas de desarrollo abiertas con el objetivo de evolucionar o mejorar ciertos aspectos de la aplicación.

- **Anexo A. Servidor Raspberry Pi. Manual de instalación y configuración:** Este anexo constituye un manual que instruye en el proceso de instalación y configuración del sistema operativo Raspbian en el ordenador de bajo coste Raspberry Pi. Posteriormente se detallan los pasos necesarios para instalar y habilitar el acceso desde redes externas al servidor de aplicaciones Apache así como la manera de habilitar el módulo PHP necesario para servir la API REST. Por último se presentan las instrucciones para implantar el servidor de bases de datos MySQL.

- **Anexo B. Manual de usuario:** En el segundo anexo de esta memoria se incluye un manual que explica e ilustra las funcionalidades de los módulos existentes en las dos aplicaciones Android desarrolladas.

- **Glosario:** Este apartado aglutina los principales términos utilizados a lo largo del presente documento, acompañados de su definición con el fin de mejorar la comprensión al lector.

- **Bibliografía:** Este apartado lista las referencias bibliográficas consultadas en la realización del proyecto.

1.4 Planificación temporal

Para explicar la planificación que se ha llevado a lo largo de la elaboración del presente proyecto, se muestran a continuación los bloques generales que agrupan cada una de las tareas realizadas:

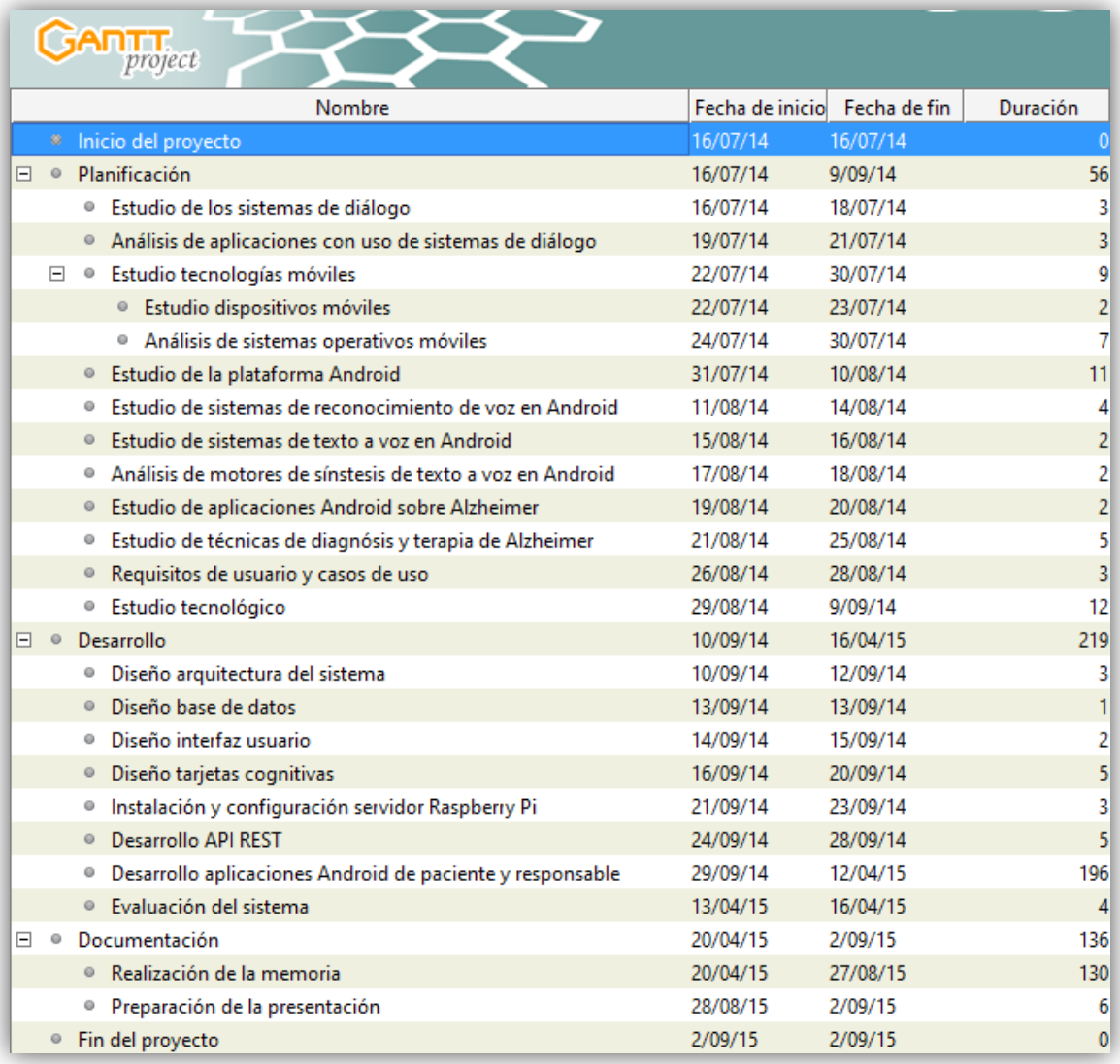
- **Planificación:** contiene el estudio de los sistemas de diálogo (incluidos los multimodales) así como un análisis de aplicaciones existentes centradas en el uso de este tipo de sistemas, el estudio de las tecnologías móviles centrado en los dispositivos móviles y un detallado análisis de los sistemas operativos móviles. También recoge el estudio de la plataforma Android con especial hincapié en sus sistemas de reconocimiento de voz y de síntesis de texto a voz (incluyendo un análisis de los principales motores disponibles). Continúa con el estudio de aplicaciones Android centradas en la enfermedad de Alzheimer así como de las técnicas de diagnóstico y terapia existentes en el ámbito médico aplicadas a esta enfermedad. Para finalizar, se exponen los requisitos de usuario y los casos de uso del sistema así como las tecnologías utilizadas para abordar los objetivos de la solución planteada.

- **Desarrollo del sistema:** agrupa el diseño de la arquitectura del sistema, de la base de datos utilizada para persistir la información, de la interfaz de usuario y de la gama de tarjetas cognitivas utilizadas en el proceso de terapia. Este bloque contiene a su vez el proceso de instalación y configuración del servidor implementado en el dispositivo de bajo coste *Raspberry Pi*, así como la implementación de la *API REST* utilizada como servicio web para interactuar con la base de datos. Por último, engloba también el desarrollo de las dos aplicaciones Android, para pacientes y responsables así como una evaluación final del sistema por diferentes perfiles de usuario.

- **Documentación del proyecto:** engloba la redacción de la presente memoria y la realización de la presentación a exponer el día de la defensa del proyecto.

Debido a la extensión del proyecto es fundamental recurrir a alguna herramienta que facilita la labor de gestionar la planificación temporal. En esta ocasión se ha recurrido a ***GanttProject*** [4] cuya principal funcionalidad reside en crear un diagrama de Gantt que muestre las fechas de inicio y fin, y por tanto la duración, de las tareas contenidas en los tres bloques principales que se acaban de describir.

La Figura 1 muestra el diagrama obtenido, siendo importante aclarar que el campo "Duración" indica el número de días naturales empleados para cada una de las tareas y/o bloques, estableciendo una dedicación media de dos horas de trabajo por día empleado.



Nombre	Fecha de inicio	Fecha de fin	Duración
Inicio del proyecto	16/07/14	16/07/14	0
Planificación	16/07/14	9/09/14	56
Estudio de los sistemas de diálogo	16/07/14	18/07/14	3
Análisis de aplicaciones con uso de sistemas de diálogo	19/07/14	21/07/14	3
Estudio tecnologías móviles	22/07/14	30/07/14	9
Estudio dispositivos móviles	22/07/14	23/07/14	2
Análisis de sistemas operativos móviles	24/07/14	30/07/14	7
Estudio de la plataforma Android	31/07/14	10/08/14	11
Estudio de sistemas de reconocimiento de voz en Android	11/08/14	14/08/14	4
Estudio de sistemas de texto a voz en Android	15/08/14	16/08/14	2
Análisis de motores de sínstesis de texto a voz en Android	17/08/14	18/08/14	2
Estudio de aplicaciones Android sobre Alzheimer	19/08/14	20/08/14	2
Estudio de técnicas de diagnósis y terapia de Alzheimer	21/08/14	25/08/14	5
Requisitos de usuario y casos de uso	26/08/14	28/08/14	3
Estudio tecnológico	29/08/14	9/09/14	12
Desarrollo	10/09/14	16/04/15	219
Diseño arquitectura del sistema	10/09/14	12/09/14	3
Diseño base de datos	13/09/14	13/09/14	1
Diseño interfaz usuario	14/09/14	15/09/14	2
Diseño tarjetas cognitivas	16/09/14	20/09/14	5
Instalación y configuración servidor Raspberry Pi	21/09/14	23/09/14	3
Desarrollo API REST	24/09/14	28/09/14	5
Desarrollo aplicaciones Android de paciente y responsable	29/09/14	12/04/15	196
Evaluación del sistema	13/04/15	16/04/15	4
Documentación	20/04/15	2/09/15	136
Realización de la memoria	20/04/15	27/08/15	130
Preparación de la presentación	28/08/15	2/09/15	6
Fin del proyecto	2/09/15	2/09/15	0

Figura 1. Planificación temporal de las tareas del proyecto

El diagrama de Gantt permite a su vez establecer relaciones de dependencia entre tareas, definiendo que tareas deben realizarse de forma previa a otra, como muestra la Figura 2. Para ello, en las propiedades de cada tarea pueden definirse un listado de tareas antecesoras. También se ha recurrido al uso de hitos, son tareas de duración cero que marcan un suceso importante en una fecha concreta, como son el "Inicio de proyecto" y "Fin de proyecto".

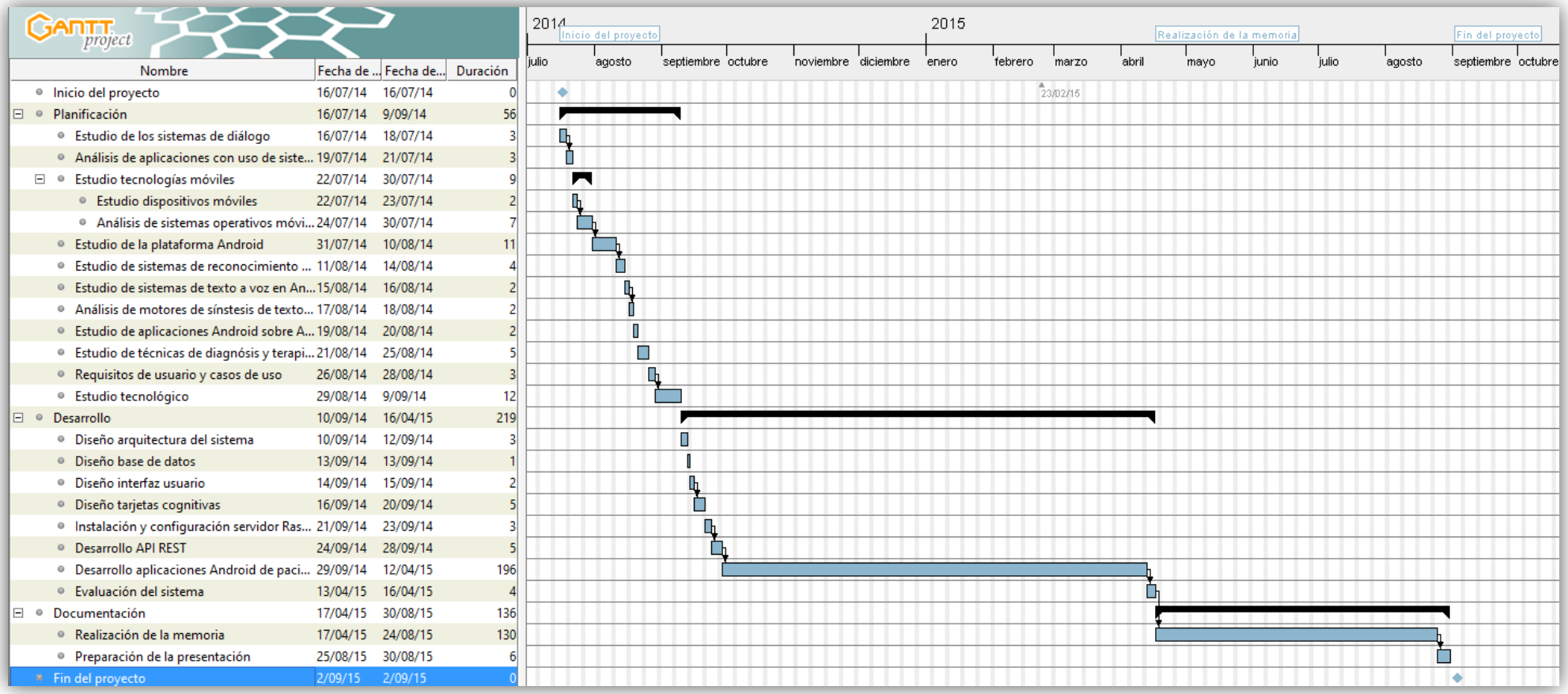


Figura 2. Diagrama de Gantt de la planificación del proyecto

En resumen, la planificación resultante indica la dedicación de un total de 411 días a la consecución del proyecto, de los cuales prácticamente la mitad se han dedicado al bloque de Desarrollo del sistema. La Planificación abarca 56 días de duración mientras que el bloque de Documentación se cierra con 136 jornadas.

1.5 Recursos

La realización del proyecto implica la adquisición y uso de una serie de recursos hardware y software, a continuación se exponen todos los elementos involucrados.

Recursos Hardware

- Ordenador portátil.
- Teléfono móvil Coolpad F1.
- Cable USB.
- Servidor Raspberry Pi.
- Tarjeta de memoria SD con 4GB de capacidad.
- Cable de red.
- Fuente de alimentación de 2 Amperios.

Recursos Software

- Entorno de desarrollo Eclipse Juno.
- Extensión ADT para Eclipse.
- Android SDK.
- JDK.
- Toad for MySQL.
- Motor de síntesis de voz de Google.
- Búsqueda por voz de Google.
- JavaMail-Android.
- Putty.
- Filezilla.
- Navegador Google Chrome.
- Extensión Postman para Chrome.
- Dropbox.
- Herramienta GanttProject.
- Servicio generador de presentaciones Prezi.
- Microsoft Office 2010.

Estos recursos tienen un coste asociado que se muestra en el siguiente apartado de “Presupuesto”.

1.6 Presupuesto

La obtención del presupuesto total del proyecto necesita del coste de los recursos humanos, los recursos hardware y software, así como los costes indirectos, como explica la plantilla de “Presupuesto de Proyecto” proporcionada por la Universidad Carlos III de Madrid, disponible en la URL:

http://portal.uc3m.es/portal/page/portal/administracion_campus_leganes_est_cg/proyecto_fin_carrera/Formulario_PresupuestoPFC-TFG%20(3)_1.xlsx

1.6.1 Costes de personal

A partir de la planificación temporal descrita en el **Apartado 1.4**, que aporta la información relativa a la duración total del proyecto (411 horas) así como la dedicación por jornada de trabajo fijada en 2 horas diarias, puede calcularse el coste de personal resultante para un único ingeniero gracias al uso de la fórmula:

Coste personal = (duración total*horas diarias)/dedicación mensual)*coste mensual por hombre

Sustituyendo los siguientes valores:

- Duración total = 411 días
- Horas diarias = 2 horas
- Dedicación mensual = 131,25 horas
- Coste mensual por hombre (ingeniero) = 2694,39 €

Se obtiene un resultado de **16.874,57 €**.

1.6.2 Costes de equipo

Los recursos mostrados en el Apartado 8.2 tienen los siguientes costes de adquisición:

Recursos Hardware

- Ordenador sobremesa y monitor →700 €
- Teléfono móvil Coolpad F1 →150 €
- Cable USB → 4 €
- Servidor Raspberry Pi → 35 €
- Tarjeta de memoria SD con 4GB de capacidad → 10 €
- Cable de red → 6 €
- Fuente de alimentación 2 Amperios → 15 €

Recursos Software

- Entorno de desarrollo Eclipse Juno → 0 €
- Extensión ADT para Eclipse → 0 €
- Android SDK → 0 €
- JDK → 0 €
- Toad for MySQL → 0 €
- Motor de síntesis de voz de Google → 0 €
- Búsqueda por voz de Google → 0 €
- JavaMail-Android → 0 €
- Putty → 0 €
- Filezilla → 0 €
- Navegador Google Chrome → 0 €

- Extensión Postman para Chrome → 0 €
- Dropbox → 0 €
- Herramienta GanttProject → 0 €
- Servicio generador de presentaciones Prezi → 0 €
- Microsoft Office 2010 → 149 €

En base a la plantilla anteriormente comentada, se realiza un cálculo de la amortización del equipo utilizado siguiendo la fórmula:

$$(A/B) \text{ x C x D}$$

Dónde:

A = Número de meses desde la fecha de facturación en que el equipo es utilizado.

B = Periodo de depreciación.

C = Coste del equipo sin IVA.

D = % del uso que se dedica al proyecto (habitualmente 100%).

El resultado puede observarse en la Tabla 1, donde el coste final de los recursos asciende a **198,27 €**.

Descripción	Coste (€)	%Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación (meses)	Coste imputable
Ordenador portátil	700	100	13,51	60	157,62
Smartphone Coolpad F1	150	100	6,57	60	16,42
Cable USB	4	100	6,57	60	0,49
Raspberry Pi	35	100	6,57	60	3,83
Tarjeta memoria SD	10	100	6,57	60	1,1
Cable de red	6	100	6,57	60	6,57
Fuente de alimentación	15	100	6,57	60	1,64
Microsoft Office 2010	149	100	4,27	60	10,60
TOTAL					**198,27 €**

Tabla 1. Amortización del material utilizado

1.6.3 Coste total

Para finalizar, además de los costes calculados anteriormente deben incluirse en el presupuesto final los costes indirectos del proyecto, calculados sobre un 20% así como un 21% de I.V.A aplicado al total. El desglose se muestra en la Tabla 2.

Descripción	Coste imputable
Costes de personal	16.874,57 €
Costes de equipo	198,27 €
Costes indirectos (20%)	3.414,56 €
TOTAL SIN IVA	20.487,41 €
IVA 21%	4.302,36 €
TOTAL CON IVA	**24.789,76 €**

Tabla 2. Coste total del proyecto

El presupuesto total de este proyecto asciende a la cantidad de **VEINTICUATRO MIL SETECIENTOS OCHENTA Y NUEVE CON SETENTA Y SEIS EUROS**.

Madrid, a 30 de Agosto de 2015

Fdo. David López Muñoz

Capítulo 2

Estado del Arte

El presente capítulo tiene como objetivo contextualizar este Proyecto Fin de Carrera. Describe la influencia que ejercen las tecnologías móviles en la sociedad, representadas a través de los dispositivos móviles así como los principales sistemas operativos que surgen para interactuar con ellos y dotarles de funcionalidad. Continúa con la justificación de elección de Android como plataforma sobre la que desarrollar la aplicación de terapia, monitorización y diagnóstico de la enfermedad de Alzheimer denominada HealthCoach, fin último del proyecto.

Posteriormente son presentados los sistemas de diálogo, junto con una descripción de los módulos que hacen posible su funcionamiento y un listado de aplicaciones tipo que ejemplifican su uso. Continúa con un estudio de las posibilidades que brinda Android a la hora de implantar servicios de reconocimiento de voz o ASR (acrónimo inglés de Automatic Speech Recognition) y síntesis de texto a voz o TTS (acrónimo inglés de Text to Speech) en el proceso de desarrollo de aplicaciones.

Para finalizar el capítulo, se realiza una introducción a la enfermedad de Alzheimer e incide en los principales procedimientos de diagnóstico y técnicas terapéuticas aplicadas a dicha demencia. Además, se presenta un listado de aplicaciones relacionadas con el Alzheimer disponibles en Google Play, que incluyen información general, procedimientos de terapia y seguimiento de la enfermedad.

2.1 Tecnologías móviles

La computación móvil ha revolucionado prácticamente todos los sectores existentes gracias a la conjunción de tres factores [2]:

- ***Hardware móvil*.** Son dispositivos de fácil transporte, descritos posteriormente en el apartado 2.1.1.

- **Software móvil.** Sistemas operativos y aplicaciones específicas para el hardware móvil. En el apartado 2.1.2 de esta memoria se realiza un estudio de los principales sistemas operativos móviles.

- **Comunicaciones móviles.** Arquitecturas de red que dotan de conexión y/o acceso a la información.

El aumento de la capacidad computacional de los dispositivos móviles, junto con sistemas operativos y aplicaciones cada vez más elaborados y unas redes móviles con acceso a Internet cada vez más rápidas y asequibles, han propiciado una gran aceptación entre los usuarios habituales de PC y portátil. En 2015, los terminales móviles han supuesto el mayor porcentaje de tráfico de datos en Internet.

La forma en la que estas tecnologías han beneficiado a diversos sectores se ejemplifica en el siguiente listado [2]:

- ***Sector comercio***. El acceso a las plataformas de ***e-commerce*** o comercio electrónico, a través de terminales móviles se denomina ***m-commerce*** e incrementa el volumen de ingresos durante el año 2014. La facilidad de acceso a la información y una experiencia de usuario personalizada con servicios como la geolocalización animan al cliente a utilizar su dispositivo móvil para realizar las compras.

- **Sector salud.** El llamado ***m-healh***, o servicio de salud a través de plataforma móvil se ha extendido rápidamente entre el colectivo médico, suponiendo la principal fuente de acceso a la información para el 51% de los doctores en España. En cuanto a los pacientes, se esperan 500 millones de usuarios de aplicaciones sanitarias móviles en 2015, con programas de monitorización, terapia o citas médicas.

- **Sector gubernamental.** En cuanto a las instituciones públicas, cada vez se desarrollan más servicios electrónicos para la gestión de trámites vía telemática. Durante el año 2012 se realizaron en España más de 500 millones de operaciones a través de su sede electrónica, alcanzando un índice de uso del 45% de la población

2.1.1 Dispositivos móviles

Existen una serie de dispositivos denominados móviles por su facilidad de manipulación y transporte, surgidos a partir del concepto inicial establecido por las PDA o Asistente Digital Personal. A continuación se describen los más frecuentes en la actualidad.

Teléfono inteligente

También conocido por el término inglés ***smartphone***, los denominados teléfonos inteligentes son dispositivos móviles de pantalla táctil y pequeño tamaño cuyas funcionalidades se han visto muy ampliadas con respecto a las de un teléfono móvil convencional. El aumento de la capacidad computacional y de su conectividad a llegado incluso a remplazar el uso de los ordenadores personales. Los smartphones supondrán un 50% del total de computadores a nivel mundial en el 2015, alcanzando el 75% en España.

Tableta

Su nombre proviene del término inglés ***tablet*** y se refiere a una computadora de escaso peso y grosor, con pantalla táctil comprendida entre 7 y 12 pulgadas de diagonal. Aunque carecen de teclado y ratón físicos, suelen soportarlos en versión inalámbrica como puede ser Bluetooth.

Tabletófono

También conocido por el termino fableta, proviene del inglés ***phablet***. Se trata de una combinación de teléfono inteligente y tableta, con un tamaño de pantalla táctil comprendido entre 5 y 7 pulgadas.

2.1.2 Sistemas operativos móviles

Partiendo de la definición propuesta por William Stallings: "*Un sistema operativo es un programa que controla la ejecución de los programas de aplicación y que actúa como interfaz entre el usuario de un computador y el hardware de la misma*" [5], se deduce la importancia de éstos para representar el nexo de unión entre máquina y usuario.

Cada día crece el número de marcas que se suman a la fabricación y distribución de dispositivos móviles, presentando productos de distintas características técnicas y calidades de fabricación. El propósito es abarcar el mayor número de compradores en relación a los requerimientos y/o presupuestos personales. A pesar de ser muy valoradas las características del hardware, como pueden ser arquitectura y velocidad del procesador, cantidad de memoria RAM, pulgadas y tecnología de la pantalla, duración de la batería, etc., existe una creciente conciencia en cuanto a la necesidad de un software depurado que exprima al máximo las capacidades hardware y proporcione una experiencia de usuario satisfactoria. En este sentido, merece la pena clasificar los dispositivos móviles en función del sistema operativo que albergan, ya que la elección de uno u otro supone acceder a funcionalidades que pueden ser propias de ese sistema.

A continuación se exponen los sistemas operativos con mayor porcentaje de implantación en la actualidad.

2.1.2.1 iOS

El sistema operativo de **Apple** para dispositivos móviles surge en 2007, junto con el lanzamiento de su primer *smartphone* denominado **iPhone**. En ese primer momento, recibió el nombre de **iPhone OS 1** y se caracterizó por la introducción de una interfaz de usuario controlada por una pantalla multitáctil, así como una pantalla principal con los iconos de las aplicaciones que apodaron *Springboard*. También es característica de esta primera versión el *Dock*, o barra inferior del *Springboard*, que contiene accesos directos a las aplicaciones que el usuario utiliza de forma más frecuente. Esta primera versión carece de soporte para aplicaciones nativas de terceros, es decir, únicamente están disponibles las aplicaciones preinstaladas por Apple en el terminal.

Con el lanzamiento en 2008 del iPhone 3G, primer dispositivo de la compañía que podía adquirirse fuera de Estados Unidos, llegaría **iPhone OS 2**. Añade soporte oficial para aplicaciones nativas desarrolladas por terceros así como la tienda denominada **iTunes App Store**, a través de la cual los usuarios pueden comprar o descargar gratuitamente las aplicaciones para disponer de ellas en su terminal, "en apenas dos meses la App Store ya contaba con más de 3.000 apps que habían sido descargadas 100 millones de veces. Un año después se convertirían en más de 85.000 apps y 2000 millones de descargas" [6]. Otra novedad de esta versión es el soporte para correos, contactos y calendarios de Microsoft mediante sincronización push[1].

Múltiples mejoras aparecerían con la tercera edición de iPhone OS en 2009, siendo una de las más solicitadas la capacidad de edición de texto mediante las acciones de cortar, copiar y pegar. Otra importante característica es la inclusión de Spotlight como elemento de búsqueda dentro de los

[1] Permite la sincronización en la aplicación cliente tan pronto como exista alguna modificación en la parte servidor. De esa manera la experiencia de usuario es completamente consistente cuando se trabaja con la misma aplicación en múltiples dispositivos [7].

contactos, calendarios, correos y aplicaciones a partir de palabras clave introducidas por el usuario. También debuta en esta versión el control de voz con soporte multilenguaje mediante el cual se pueden indicar ciertas acciones al dispositivo como por ejemplo "llamar a David". En la actualización **iPhone OS 3.2** surgió el iPad 1, un iPhone de "pantalla grande" con 9,7 pulgadas de diagonal y dos versiones, una con conexión 3G[2] a datos de redes de telefonía sin posibilidad de llamadas, y otra únicamente con conexión a redes Wi-Fi.

La evolución hacia la actual denominación del sistema operativo surgió en 2010 con el lanzamiento conjunto de **iOS 4** y iPhone 4. Supuso la mayor evolución entre versiones gracias a las nuevas características y **APIs** a disposición de los desarrolladores (acrónimo inglés de *Application Programming Interface*), además del hecho de comenzar a distribuirse en la gama de dispositivos iPhone, iPad e iPod Touch. La multitarea supuso un avance muy importante en la experiencia de usuario de iOS 4, ya que permitió el uso de varias aplicaciones simultáneas, a pesar de sólo disponer de una aplicación en primer plano las demás prosiguen su ejecución en segundo plano. Otra característica importante fue la creación de carpetas que permiten organizar de manera más eficiente las aplicaciones permitiendo clasificaciones temáticas.

De la mano de **iOS 5** y el nuevo smartphone iPhone 4S, aparece en 2010 **Siri** como un asistente personal inteligente que supone la evolución del control por voz, introducido en iOS 3. Siri no se limitaba a realizar acciones concretas, sino que supuso la interacción entre el usuario y el dispositivo móvil mediante lenguaje natural. También se presentó el centro de notificaciones (inspirado por Android) al que se accede deslizando el dedo sobre la pantalla desde la parte superior hacia abajo. En él se presentan al usuario una serie de eventos lanzados por las aplicaciones a modo de alertas, con el fin de presentar información útil referente al uso del dispositivo, como puede ser la llegada de un correo, la hora de activación de una alarma, etc. Merece la pena destacar también que Apple decide eliminar la necesidad de sincronización mediante cable a un ordenador tanto para la activación inicial del dispositivo como para la transferencia de datos, añadiendo de esta forma capacidades inalámbricas de sincronización.

Con **iOS 6** y iPhone 5 se produjo en septiembre de 2012 un drástico reemplazo de Google Maps por Apple Maps, versión propia desarrollada por Apple en colaboración con la plataforma de navegación TomTom. También se obtuvo una integración en el sistema operativo de la red social Facebook, permitiendo por ejemplo actualizaciones de estado desde Siri.

El año 2013 conllevó una evolución en cuanto al diseño de la interfaz de usuario, presentado con **iOS 7** y los dispositivos iPhone 5C y 5S. Predomina el diseño minimalista con iconos planos y transparencias, además de un centro de control renovado al que se accede de forma contraria al centro de notificaciones, es decir, deslizando el dedo por la pantalla desde la parte inferior hacia arriba. Desde él se pueden acceder a opciones de conectividad como Bluetooth o Wi-Fi, iluminación de la pantalla, volumen y vibración del terminal, cámara, etc. En esta versión se optimiza la multitarea y se automatiza el proceso de actualización de las aplicaciones. Touch ID supone un sistema biométrico de acceso al terminal mediante la huella del propietario.

Con la presentación de **iOS 8**, y los modelos iPhone 6 y iPhone 6 Plus, llegamos al sistema operativo vigente desde septiembre de 2014 hasta el momento de redactar esta memoria. De este último podemos destacar la funcionalidad que han denominado *Handoff*, consistente en la continuidad de acciones entre dispositivos iPhone, iPad y Mac, gracias a la cuál puede por ejemplo comenzar a

[2] Redes de tercera generación con tecnologías de acceso UMTS/HSDPA (850, 1900 y 2100 MHz) [8].

redactarse un correo desde cualquiera de ellos para continuar más tarde desde otro, recuperando el punto exacto de la redacción dónde se había dejado. De igual manera, es factible recibir y contestar una llamada telefónica, o redactar un mensaje de texto desde un Mac. El único requisito para utilizar esta interoperabilidad es que todos los dispositivos estén registrados con la misma cuenta de iCloud, que es el sistema de almacenamiento en la nube de Apple. También debuta una nueva aplicación denominada Salud [9] que recopila todo tipo de información referente a la forma física y salud proporcionada por otras aplicaciones mediante la nueva API *HealthKit*. También se añade la posibilidad de instalar aplicaciones de teclado de terceros, como pueden ser *Swiftkey* o *Swype*, hasta ese momento sólo era posible utilizar el proporcionado por Apple.

Los desarrolladores que deseen realizar sus aplicaciones para iOS están obligados a pagar una membresía anual de 99 dólares, de cara a utilizar los entornos de desarrollo proporcionados por Apple así como poder publicar en la AppStore. Estos entornos únicamente pueden instalarse en equipos MAC.

2.1.2.2 Android

Es el sistema operativo desarrollado por Google junto a un conjunto de empresas del sector tecnológico denominado Open Handset Alliance, con el propósito de crear plataformas abiertas para teléfonos móviles. Basado en el *kernel* de Linux, existe una versión denominada Android Open Source Project (AOSP) sin añadidos y personalizaciones de los fabricantes que lo instalan en sus dispositivos.

Aunque presentado oficialmente en versión beta en noviembre de 2007, no sería hasta el año siguiente cuando la primera versión estable viera la luz junto con la comercialización del primer dispositivo soportado.

La evolución del sistema operativo lo ha llevado a ampliar el tipo de terminales soportados más allá de los teléfonos móviles y tabletas, existen productos para televisión con Android TV, automóviles con *Android Auto* y relojes inteligentes con *Android Wear*.

A modo de resumen se describe cada una de las versiones del sistema operativo para obtener una vista global de la evolución e historia de Android.

Android 1.0 (Apple Pie)

El 23 de septiembre de 2008 Google presenta el terminal HTC Dream/T-Mobile G1 junto con la versión 1.0 de Android, con nombre en código Apple Pie. Supone el punto de partida del sistema operativo, con algunas características que han perdurado a lo largo de las versiones posteriores, como son:

- **Android Market**. Es la tienda de aplicaciones y servicios propia de Android, actualmente llamada Google Play. También gestiona el proceso de actualización de las aplicaciones.
- **Gestor de notificaciones**. Las notificaciones del sistema y aplicaciones aparecen en la barra de estado.
- **Aplicaciones Google**. Incorpora una serie de aplicaciones nativas para consumir servicios de Google, como Google Maps, Google Search para búsquedas en internet, aplicaciones del dispositivo, contactos, calendario, etc., Google Sync para sincronizar Gmail, contactos y calendario, Google Talk para mensajería instantánea y Youtube.

- **Navegador web**. Permite visualizar múltiples páginas HTML y XHTML mediante una interfaz de tarjetas o multiventana.
- **Marcación por voz**. Permite marcar y llamar sin utilizar el teclado virtual.
- **Organización mediante carpetas**. Permiten agrupar varios iconos de aplicaciones en el escritorio.
- **Cámara fotográfica**. Es un soporte inicial sin soporte para cambio de resolución, calidad, balance de blancos, contraste, etc.
- **Wi-Fi y Bluetooth**. Soporte para las tecnologías de conectividad más extendidas.
- **Reproductor multimedia**.
- **Pantalla de inicio y personalización**. Gestionada por la aplicación encargada de representar la interfaz de usuario, denominada *launcher*. En esta primera versión, se habilita la elección del fondo de pantalla y el soporte para widgets[3] en el escritorio.
- **Mensajes de texto y MMS**.

Android 1.1 (Banana Bread)

Esta actualización es lanzada en exclusiva para el HTC Dream en febrero de 2009. Además de corregir ciertos problemas, supone un cambio en la versión de la API e incorpora novedades funcionales como:

- Archivos multimedia adjuntos en los mensajes.
- Reseñas e información de usuario en Google Maps.
- Pantalla de llamada en modo manos libres más amplia y posibilidad de esconder el marcador numérico.

Gráficamente no supone ninguna evolución en la interfaz de usuario.

Android 1.5 (Cupcake)

El 30 de abril de 2009 se presenta Android 1.5 con importantes cambios en la interfaz de usuario así como características sociales:

- Redes sociales. Se habilita la subida de vídeos a Youtube y fotografías a Picasa.
- Elección de teclado. A partir de esta actualización se añade el soporte para teclados de terceros.
- Widgets mejorados. Posibilidad de instalación de widgets más visuales.
- Copiar y pegar. Añadida la funcionalidad de copiar y pegar texto desde el navegador.
- Transiciones animadas y rotación automática de pantalla.
- Framework reconocimiento voz. Se añade soporte para el uso de las librerías de reconocimiento de voz.

[3] Son versiones en miniatura de las aplicaciones que se muestran dentro de otras aplicaciones, como la pantalla de inicio de Android, para simplificar el acceso a la información.

Android 1.6 (Donut)

Meses después del anuncio oficial de Android 1.5, concretamente el 15 de septiembre de 2009, es presentado Android 1.6 con las siguientes novedades:

- **Motor TTS**. Se añade un módulo *text-to-speech* (síntesis de texto a voz) a disposición de cualquier aplicación del sistema operativo.
- **Evolución de búsqueda por voz y teclado**. Incluye la posibilidad de búsquedas en la web, historial, favoritos, etc., a través del módulo reconocedor de voz o entrada de texto.
- Entrada de gestos. Nueva api de creación y reconocimiento de gestos.
- Kernel 2.6.29 de Linux.

Android 2.0/2.1 (Eclair)

La llegada de Android Eclair en octubre de 2009, a pesar de basarse en el mismo *kernel* que Android 1.6, trae una serie de mejoras como son soporte multicuenta para correo electrónico, software de gestión de cámara incluyendo flash, zoom, balance de blancos, enfoque, etc., escritura en el teclado virtual más rápida con diccionario inteligente, navegador de serie actualizado para soportar HTML5 y zoom, Google Maps mejorado, interfaz de usuario renovada con fondos de pantalla animados y una importante optimización de los controladores de hardware.

Hasta esta versión, la entrada de texto por voz únicamente era posible en las búsquedas de la aplicación Google Search, pero Eclair introduce el dictado de voz (ver **Apartado 2.4.2**) para cualquier campo de texto de las aplicaciones, también conocido como dictado por voz.

Android 2.2 (Froyo)

La versión 2.2 se lanza en mayo de 2010 basándose en una versión del núcleo de Linux, concretamente el 2.6.32. El rasgo más destacable son las numerosas optimizaciones del sistema operativo para obtener una experiencia de usuario mejorada en cuando a rendimiento, principalmente gracias al nuevo método de compilación Just-in-Time (JIT).

Se añade también la posibilidad de crear una red Wi-Fi compartiendo la conexión del terminal (*tethering*) denominada Wi-Fi hotspot, soporte JavaScript y Adobe Flash en el navegador, notificaciones push mediante el servicio Google Cloud to Device Messaging (G2CM), soporte para pantallas de alta densidad de puntos por pulgada (PPI).

Una novedad importante son las llamadas *Voice Actions* o Acciones de Voz en castellano, representan una nueva forma de gestionar el teléfono mediante órdenes por voz.

Android 2.3 (Gingerbread)

El lanzamiento de Android 2.3 (diciembre de 2010) se acompañó de varias novedades importantes, como soporte para el protocolo SIP de telefonía VoIP, soporte para la tecnología inalámbrica NFC, nuevos sensores como giroscopio y barómetro, nuevo gestor de descargas optimizado para transferencia de archivos grandes, soporte multicámara y nuevos *códecs* multimedia como WebM/VP8 y AAC, mejoras en la precisión del teclado y entrada por voz. Cuenta además con un diseño de interfaz más simple y eficiente en cuanto a consumo de recursos.

Android 3.0/3.1/3.2 (Honeycomb)

Las versiones 3.X de Android surgen en febrero de 2011 con el propósito lanzar una versión optimizada de Android para tablets, por lo que no está disponible para ningún teléfono móvil. Incluye un teclado optimizado para pantallas grandes, aplicaciones de correo, contactos y navegador adaptados, barra de notificaciones en la parte inferior de la pantalla, nueva interfaz para copiar-pegar texto, ajuste de tamaño de widgets y una serie de mejoras en la interfaz para hacerla más intuitiva.

Android 4.0 (Ice Cream Sandwich)

En el salto a la versión 4.0 de Android en octubre de 2011 se incorpora el kernel 3.0.1 de Linux. Constituye una versión fundamental al introducir compatibilidad con tabletas y teléfonos inteligentes, que será conservada en todas las versiones posteriores hasta la fecha.

Incorporó otras características como capturas de pantalla, acceso a aplicaciones desde la propia pantalla de bloqueo, versión del navegador Google Chrome para Android con sincronización de marcadores y un límite de quince pestañas simultáneas, creación de carpetas mediante arrastrar y soltar, dictado de voz mejorado, rediseño de la galería de imágenes, así como las habituales mejoras de rendimiento y estabilidad.

Android 4.1/4.2/4.3 (Jelly Bean)

Jelly Bean es presentado el 9 de julio de 2012 con el propósito de superar los problemas de rendimiento y batería de su versión predecesora. Google adoptará una serie de medidas dentro del que denominó “Proyecto Butter” para conseguir una experiencia de usuario suave y fluida.

En esta versión se inaugura, como parte de Google Search, el asistente personal inteligente que recopila nuestros hábitos, gustos y búsquedas recientes para presentarnos información de forma anticipada en forma de tarjetas, denominado Google Now. Integra además una nueva versión de Voice Search con el añadido de “OK Google”, función por la que se pueden realizar búsquedas de voz sin pulsar si quiera el icono del micrófono, bastan con pronunciar esas dos palabras para que el sistema interprete las que precedan como los términos o indicaciones de búsqueda.

Otras funciones destacables son la impresión inalámbrica, nueva entorno de ejecución denominado ART, una evolución experimental de Dalvik, y soporte con el dispositivo Google Chromecast.

Android 4.4 (KitKat)

El 31 de octubre de 2013 se presenta Android 4.4, con importantes características como mayor rendimiento del sistema incluso para dispositivos con recursos limitados, mejora de la compatibilidad de ART y de la multitarea, uso de Chrome en las vistas web de las aplicaciones, evolución de la aplicación de descargas y correo electrónico, modo inmersivo (se ocultan las barras de estado y navegación) en aplicaciones stock, así como nuevos efectos visuales.

Android 5.0 (Lollipop)

La versión más reciente de Android en el momento de redactar esta memoria, es presentada el 3 de noviembre de 2014, bajo la denominación de Android 5.0. La interfaz gráfica ha sido rediseñada para conseguir un aspecto más sencillo y plano, denominado "*Material design*". El nuevo patrón de diseño no sólo se limita al sistema y las aplicaciones preinstaladas, sino que Google pretende sea el nuevo patrón de diseño para todos los desarrolladores.

Otra novedad es el renovado sistema de notificaciones con una nueva plantilla que permite hasta seis elementos de control de notificaciones con funciones como responder mensajes desde la pantalla de bloqueo, clasificación inteligente, etc. Se obtienen también importantes las mejoras de rendimiento gracias al entorno de ejecución ART que ahora es el utilizado por defecto, soporte para plataformas de 64 bits, así como un sistema de gestión de la batería más inteligente que promete extender hasta 90 minutos la autonomía y las frecuentes mejoras de estabilidad.

El 9 de marzo de 2015 Google lanza la versión 5.1 con mejoras de rendimiento, soporte multi-sim y sistema de protección de terminales permitiendo bloquearlos en caso de robo o extravío.

Como último apunte sobre Android, existe una marcada fragmentación de versiones en activo. En la Figura 3 se observan los datos obtenidos a 4 de mayo de 2015. Predominan las versiones 4.X, es decir tanto KitKat como JellyBean, sin embargo a Lollipop le queda bastante camino por recorrer para estar tan extendido como en sus versiones predecesoras

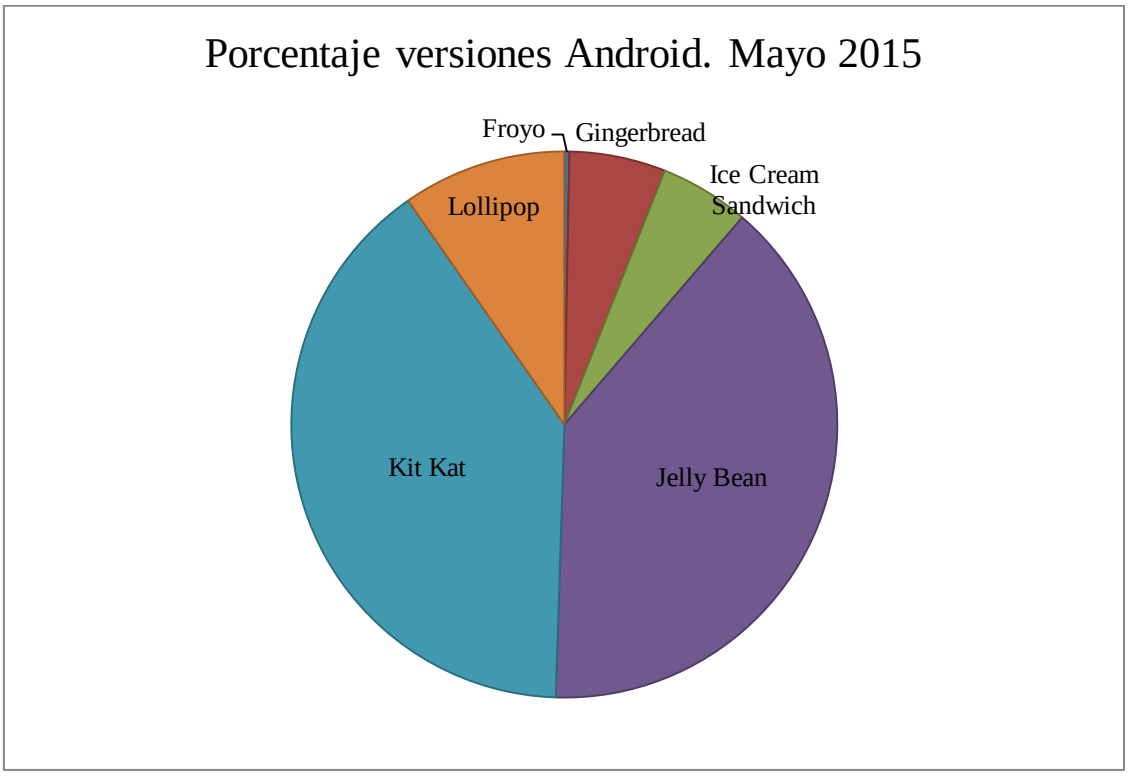


Figura 3. Fragmentación de versiones Android a fecha Mayo de 2015

2.1.2.3 Windows Phone

Windows Phone (WP) es el sistema operativo móvil desarrollado por Microsoft, anteriormente llamado Windows Mobile. Su núcleo se basa en Windows CE 6.0 en las versiones tempranas para posteriormente evolucionar a un núcleo Windows NT.

La novedosa interfaz diseñada por Microsoft, denominada Modern UI (anteriormente Metro), se basa en una pantalla de inicio compuesta de unos "azulejos" geométricos de colores básicos denominados Live Tiles (Figura 4). En esencia, son iconos anclados en la pantalla principal que representan enlaces a aplicaciones, música, contactos, el tiempo, etc. Poseen además la capacidad de representar información en tiempo real con animaciones.

Figura 4. Pantalla de inicio en Windows Phone 8.1 con Live Tiles [10]

Destaca el uso de Bing como motor de búsqueda, en un primer momento reemplazable por Google pero suprimida la posibilidad en versiones posteriores, e Internet Explorer como navegador predefinido.

La primera versión, denominada **WP 7.0**, ve la luz en octubre de 2010 con déficits a nivel de rendimiento y funcionalidad del sistema. Pasados unos meses (marzo de 2011) se realiza el lanzamiento de **WP 7.1**, con importantes mejoras de rendimiento y optimización del tiempo de arranque. Las siguientes versiones denominadas **WP 7.5.0** y **WP 7.5.1** añaden multitud de mejoras, algunas tan importantes como la multitarea y una rebaja de los requisitos mínimos de hardware necesarios para instalar WP (a petición de Nokia). La versión 7.8, que sería la última con esta numeración, supone una transición hacia una interfaz de usuario similar a la que tendría la versión 8.

Windows Phone 8.0 supone una evolución en el núcleo del sistema, comenzando a utilizar Windows NT. Desde finales de 2012, pasará por tres subversiones hasta llegar a la 8.0.3, añadiendo gradualmente en cada una de ellas nuevas mejoras, como NFC, Internet Explorer 10, Skype, WiFi en

suspensión, radio FM, optimización de HTML5, modo conducción, soporte para procesadores de cuatro núcleos, etc.

En abril de 2014 se produce el lanzamiento de **WP 8.1**, versión que perdura hasta la fecha de realización de este proyecto. Supone un cambio de núcleo del sistema pasando a utilizar Windows. En ella, se incluyen una serie de mejoras para intentar rivalizar con sus inmediatos competidores, Android e iOS, los cuales siguen ocupando el primer y segundo puesto respectivamente en cuanto a cuota de mercado. A continuación se muestran las más importantes:

- **Cortana**. Es el asistente virtual que debuta con esta versión para competir con los asistentes Siri y Google Now. El usuario puede interactuar con Cortana utilizando lenguaje natural, sin necesidad de pronunciar comandos concretos, gracias al uso de los servicios proporcionados por el buscador Bing. A los ya habituales usos de gestión de llamadas y alarmas, búsqueda de información en la web, etc., se une una característica novedosa denominada Cuaderno de Cortana. Esta función recopila información personal del usuario, como sus intereses, gustos y acciones que suele solicitar, permitiendo editar o eliminar cada una de las entradas. Con el paso del tiempo, escapa de combinar la información guardada en el Cuaderno con otra relevante para poder realizar sugerencias proactivas. La interfaz de Cortana se muestra en la Figura 5.
- **Centro de notificaciones**. Desde esta versión es posible, al estilo Android o iOS, deslizar desde la parte superior de la pantalla para acceder al centro de notificaciones. Desde él se pueden consultar los eventos susceptibles de notificación o acceder a opciones de configuración rápida como habilitar/deshabilitar el WiFi o Bluetooth.
- **Personalización pantalla de inicio**. Permite la elección de fondo de pantalla y una tercera columna de iconos o Live Tiles.
- **Sensores WiFi, datos, almacenamiento y batería**. Se tratan de aplicaciones destinadas a la monitorización y optimización de los recursos.
- **Nuevas aplicaciones preinstaladas**. Como son Comida y Bebida, Viajes y Mapas, Salud y Ejercicios, y Mapa, que incluye una licencia gratuita con todos los mapas mundiales gracias al acuerdo con HERE Drive+. La distribución de aplicaciones se realiza a través de su tienda de nominada Windows Phone Store [11].
- **Word Flow**. Es el nuevo teclado por defecto para WP cuya principal novedad es la posibilidad de escribir deslizando el dedo por la pantalla seleccionando las letras que conforman la palabra a obtener.

La distribución de aplicaciones para terminales con Windows Phone se realiza a través de la Store. Esta tienda está gestionada por Microsoft, quien se encarga de examinar y validar el contenido adecuado de las aplicaciones, juegos, vídeo y música subidos. En enero de 2015 dispone de más de 560.000 aplicaciones, con un crecimiento mensual cercano a 15.000 aplicaciones nuevas. En cuanto a las actualizaciones de software, se sirven de Microsoft Update al igual que el resto de sistemas Windows.

De cara al desarrollador, el acceso a los entornos de desarrollo necesarios para realizar aplicaciones compatibles es gratuito, sin embargo, la publicación en el Store tiene un coste asociado de 19 dólares al año para desarrolladores individuales o bien 99 dólares anuales para compañías.

Figura 5. Interfaz de asistente virtual Cortana, disponible desde Windows Phone 8.1 [12]

2.1.2.4 Blackberry OS

Blackberry Limited, anteriormente Research In Action (RIM), es la encargada de desarrollar las distintas versiones del sistema operativo Blackberry OS, incluido en su gama de smartphones Blackberry.

Bajo licencia propietaria y con un núcleo basado en la máquina virtual de Java en sus primeras versiones, el sistema operativo se caracteriza por la capacidad multitarea y una interacción del usuario con la interfaz a través de una rueda de desplazamiento o *trackwheel*, bola de desplazamiento o *trackball*, panel táctil o *touchpad*, o pantallas táctiles. La aparición del trackball supuso la posibilidad de desplazamiento en diagonal. Otra característica diferenciadora de la gama Blackberry es la existencia de un teclado físico QWERTY en la mayoría de sus dispositivos, que facilita la navegación web y la gestión de los correos electrónicos sincronizados con cuentas Microsoft Exchange, Novell GroupWise o Lotus Note. Estas funciones han propiciado una gran aceptación para el uso profesional de los dispositivos Blackberry.

Desde el lanzamiento de la versión 1.0 de Blackberry OS en enero de 1999, el sistema operativo y los dispositivos de la gama Blackberry han sufrido una fuerte evolución hasta llegar a las versiones más recientes. El 30 de enero de 2013 se lanza **Blackberry 10** sin utilizar como base ninguna versión anterior, es decir, es desarrollado desde cero utilizando como base el sistema operativo QNX. La interfaz de usuario sufre una actualización, se añade un teclado inteligente y una multitarea real en las aplicaciones. También incorpora un centro de notificaciones llamado Hub, donde se lista información referente a recepción de correo electrónico, redes sociales, etc., que son seleccionables para acceder a la aplicación que las genera. La inclusión de gestos en Blackberry OS 10 facilita la navegación y acceso a ciertas funciones al deslizar el dedo en la pantalla en una dirección y sentido concretos.

Quizás una de las novedades más importantes de esta versión sea la capacidad de ejecutar aplicaciones nativas Android, gracias a una modificación en la arquitectura del sistema para incluir una adaptación del entorno de ejecución de aplicaciones Android (Dalvik).

Blackberry 10.3.1 supone una importante evolución al incorporar la tienda de aplicaciones de Amazon, lo que aumenta de manera considerable el catálogo ya existente en la tienda Blackberry World. Aparece también Blackberry Blend, funcionalidad mediante la que puede accederse a las notificaciones, mensajes, contactos, galería multimedia y otra serie de información del dispositivo móvil desde un ordenador o tablet. En relación a la anterior, Blackberry Link permite sincronizar música, fotos, vídeos y cualquier archivo multimedia entre un dispositivo Blackberry y un PC o Mac. También introduce un asistente personal virtual, al estilo Siri de iOS o Google Now de Android, con el que interactuar por medio de texto o voz, en lenguaje natural.

El desarrollo de aplicaciones por parte de terceros se realiza a través de las APIs proporcionadas por Blackberry para C/C++/Qt, JavaScript/CSS/HTML, ActionScript/AIR o Java (Android). Es gratuito tanto el acceso a las herramientas de desarrollo y a la publicación de las mismas. Sin embargo, es necesaria una firma digital de desarrollador si la aplicación accede a ciertas funcionalidades del sistema.

2.1.2.5 Firefox OS

Firefox OS es el sistema operativo móvil de código abierto creado por Mozilla Corporation. Lanzado en 2012, es mantenido por una comunidad de desarrolladores sin ánimo de lucro.

Su arquitectura se basa en tres componentes principales (Figura 6) descritos a continuación:

Gaia: interfaz gráfica de usuario

Implementada en HTML5, CSS y JavaScript, esta interfaz de usuario se comunica con los niveles inferiores del sistema operativo únicamente a través de web APIs. La gestión de la pantalla principal, pantalla de bloqueo y aplicaciones web preinstaladas en el dispositivo se realiza desde esta capa de abstracción.

Gecko: entorno de ejecución

Se trata del mismo entorno de ejecución utilizado por el navegador Firefox, encargado de tratar y renderizar el contenido web. Su cometido es hacer que todas las APIs funcionan correctamente en cada uno de los sistemas operativos soportados por **Gecko**.

Gonk: núcleo y drivers

Es el núcleo del sistema operativo, consistente en un kernel de Linux basado en Android Open Source Project (AOSP), con el añadido de los drivers específicos de los componentes del dispositivo como el bluetooth, cámara, módulo GPS, etc.

Figura 6. Componentes de la arquitectura de Firefox OS

La distribución de las aplicaciones de terceros se realiza a través de la tienda estrenada junto con el lanzamiento de **Firefox OS**, denominada **Firefox Marketplace** [13]. La tienda se encuentra clasificada por categorías, y existe un sistema de puntuación y comentarios que fomenta la colaboración en las opiniones de los usuarios. La instalación y/o actualización de las aplicaciones es prácticamente inmediata ya que la mayoría de ellas están alojadas en la web.

De cara al desarrollador, no tiene ningún coste asociado el desarrollo o publicación de las aplicaciones. Los únicos requerimientos para crear una aplicación son:

- Navegador Firefox actualizado a la última versión existente.
- Editor de texto plano. Servidor local o remoto para alojar los archivos de la aplicación.

En esencia, las aplicaciones para Firefox OS son aplicaciones web basadas en HTML5, CSS3 y JavaScript, únicamente es necesario un manifiesto de aplicación que indique se trata de una versión compatible con el sistema operativo móvil, habilitando el acceso a los recursos hardware del dispositivo y permitiendo la publicación en el Marketplace. Dos características que diferencian una aplicación bien adaptada a este sistema operativo, son la capacidad de trabajo online y la optimización en cuanto a consumo de batería.

2.1.2.6 Ubuntu Phone OS

Ubuntu Phone, también conocido como Ubuntu Touch, es el sistema operativo móvil con núcleo Linux, diseñado y desarrollado por Canonical. Fue presentado a la comunidad a principios de 2013 y distribuido bajo licencia GNU GPL.

La interfaz gráfica, Unity, representa el esfuerzo de Canonical para conseguir un elemento común para ordenadores, móviles y tablets, basada en el framework Qt 5. En los dispositivos táctiles, existe una pantalla principal desde la que podemos acceder a ciertas funciones realizando gestos táctiles. Si deslizamos el dedo desde la parte superior de la pantalla hacia abajo se accede a las notificaciones y opciones del teléfono (volumen, vibración, brillo de pantalla, Wi-Fi, etc.). Al deslizar desde el extremo izquierdo de la pantalla hacia la derecha se muestra el lanzador de aplicaciones, donde se encuentran las aplicaciones o características más utilizadas por el usuario. Deslizando desde el extremo derecho hacia la izquierda presenta una nueva pantalla con todas las aplicaciones corriendo en segundo plano,

si se selecciona cualquiera de ellas se reanudará el foco de dicha actividad. Por último, al deslizar desde la parte inferior de la pantalla hacia arriba se muestran los controles.

El desarrollo de aplicaciones para Ubuntu Phone OS se realiza a través de HTML5 o QML (Qt Meta Language). En la plataforma de aplicaciones de Ubuntu (Ubuntu App Platform) se encuentra información relativa al SDK (Software Development Kit) de Ubuntu, el cuál dispone de APIs específicas si se desea desarrollar en QML [14] o bien HTML5 [15]. Estas APIs proveen acceso a características físicas del dispositivo móvil, como los sensores, así como a la interfaz gráfica. Canonical pretende con la filosofía de las "Ubuntu apps" el desarrollo multiplataforma, un único desarrollo es compatible con todas las plataformas que soporta (teléfonos, tabletas, ordenadores, etc.). La distribución de las aplicaciones, tanto nativas como web, se realiza a través de la tienda denominada Ubuntu Store, la cual a día de hoy dispone de un catálogo reducido, sin el apoyo de aplicaciones de peso como el programa de mensajería Whatsapp.

En la actualidad existen únicamente dos dispositivos que alojan este sistema operativo, el fabricado por la empresa española BQ, denominado Aquaris E4.5 Ubuntu Edition [16] y el desarrollado por la empresa china Meizu, el MX4 Ubuntu edition.

2.2 Justificación elección SO

Son múltiples los factores y características que diferencian entre sí los sistemas operativos móviles. De cara a la implementación de la aplicación de este PFC, se han seleccionado aquellos criterios que se consideran críticos para acometer la elección final de la plataforma:

- **Cuota de mercado actual y futura.**
- **Número de aplicaciones y desarrolladores.**
- **Costes de desarrollo y publicación.**

Cuota de mercado actual y futura

El número de dispositivos vendidos en cada plataforma indica el nivel de aceptación y expansión del sistema operativo. Desarrollar una aplicación para un SO con una gran cuota de mercado asegura poder distribuirla a un mayor número de personas.

Según estadísticas del IDC (International Data Corporation) a nivel mundial [17], Android lidera el ranking de ventas de smartphones durante el año 2014, con un volumen de 1.059 millones de unidades vendidas, representando un 81,5% de la cuota de mercado durante ese año. El segundo puesto lo ocupa iOS con una cuota de mercado en 2014 del 14,8%, sufriendo una ligera descenso en las ventas con relación al 2013. Ambas plataformas abarcan la práctica totalidad de las ventas sumando un 96,3%. Relegado en la distancia, el tercer lugar lo ocupa Windows Phone con un 2,7% de cuota de mercado. Blackberry obtiene la cuarta posición con un simbólico 0,4%, sufriendo un decrecimiento con respecto al año anterior del 69,8%. El porcentaje restante de la cuota de mercado para el 2014 lo ocupa un conjunto de sistemas operativos emergentes, con un crecimiento anual con respecto a 2013 del 234,8%, aunque de momento minoritarios (0,6% de cuota de mercado conjunta). Lo conforman Ubuntu Phone, Firefox OS, Tizen OS y Sailfish OS. Estos datos pueden consultarse en la Tabla 3.

Sistema operativo	2014 Volumen ventas (millones)	2014 Cuota mercado	2013 Volumen ventas (millones)	2013 Cuota mercado	2013-2014 Crecimiento anual
Android	1,059.3	81.5%	802.2	78.7%	32.0%
iOS	192.7	14.8%	153.4	15.1%	25.6%
Windows Phone	34.9	2.7%	33.5	3.3%	4.2%
BlackBerry	5.8	0.4%	19.2	1.9%	-69.8%
Otros	7.7	0.6%	2.3	0.2%	234.8%
Total	1,300.4	100.0%	1,018.7	100.0%	27.7%

Tabla 3. Volumen de ventas, cuota de mercado y crecimiento anual en 2014 con respecto a 2013 de las cuatro principales plataformas móviles [17]

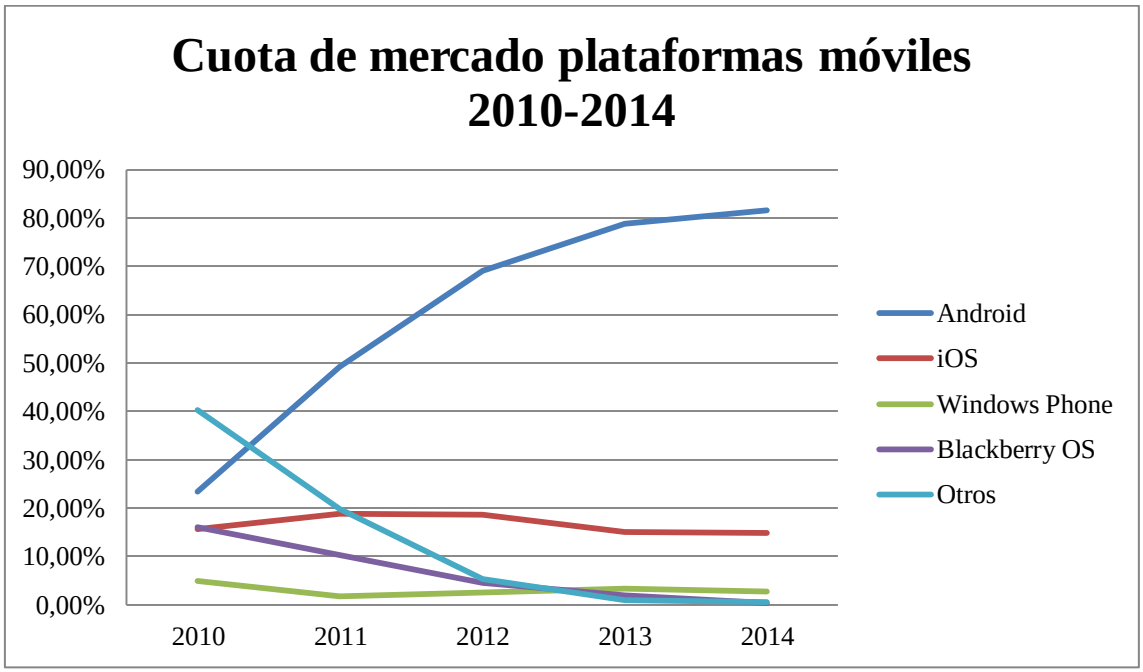


Figura 7. Evolución de la cuota de mercado por sistema operativo móvil en el período 2010-2014 [18]

Continuando con los estudios del IDC a nivel global [18], pronostican una caída del crecimiento de ventas anual del 8,3% en 2017 y del 6,2% en 2018. La razón principal es la saturación de terminales distribuidos en los principales mercados mundiales, quedando como potenciales nichos los emergentes como India. En la Tabla 4 se observa el pronóstico de la cuota de mercado en 2018 desglosado por plataforma, siendo destacable el crecimiento de Windows Phone y sobre todo el conjunto de sistemas operativos emergentes, con cifras que rondan el 30%. No menos destacable es el descenso de ventas de Blackberry, que retrocede un 22,6%.

Sistema operativo	2018 Pronóstico volumen ventas (millones)	2018 Pronóstico cuota mercado	2014-2018 Pronóstico crecimiento
Android	1,321.1	76.0%	10.7%
iOS	249.6	14.4%	10.2%
Windows Phone	121.8	7.0%	29.5%
BlackBerry	5.3	0.3%	-22.6%
Otros	40.7	2.3%	32.7%
Total	1,738.5	100.0%	11.5%

Tabla 4. Volumen de ventas, cuota de mercado y crecimiento anual en 2014 con respecto a 2013 de las cuatro principales plataformas móviles

Como consecuencia directa de la saturación de los mercados, los principales fabricantes de dispositivos móviles intentan minimizar la caída de ventas mediante un descenso de precios, lo que a su vez permite una mayor aceptación en las naciones emergentes. En 2013 se vendieron 322,5 millones de smartphones con un coste inferior a 150 dólares. En la Tabla 5 puede apreciarse la tendencia comentada, gracias a la información desglosada con la media de precios en función del sistema operativo. Contrasta la diferencia del precio medio de las dos plataformas más extendidas, Android e iOS, teniendo de media un sobrecoste de 400 dólares los terminales de Apple.

Sistema operativo	2014 Precio medio de venta smartphone (dólares)	2018 Estimación precio medio venta smartphone (dólares)	2014-2018 Crecimiento
Android	247	202	-6.1%
iOS	649	610	-1.2%
Windows Phone	265	195	-8.3%
BlackBerry	339	324	-3.8%
Otros	154	173	1.4%
Total	**308**	**260**	**-5.0%**

Tabla 5. Representación de los precios medios de venta de los smartphones en función de la plataforma

Número de aplicaciones y desarrolladores

Un factor importante a la hora de elegir plataforma sobre la que desarrollar, es conocer el número de aplicaciones existentes en cada una de las tiendas de distribución digital, generalmente denominadas ***markets*** o ***stores***, pero también el número de desarrolladores involucrados. La Figura 8, basada en datos de enero de 2015 [19], establece al store de Android (Play Store) la primera posición con una cifra cercana al millón y medio de aplicaciones, aventajando en 220.000 las disponibles en el App Store de iOS. Muy lejos quedan las cifras del Windows Phone Store, Amazon Appstore y Blackberry World respectivamente. Aunque el market de Amazon aloja aplicaciones Android, éstas

también pueden ser instaladas en los últimos dispositivos Blackberry, por lo que su catálogo se sumaría al propio de la marca, obteniendo una suma total superior a las disponibles en el Windows Phone Store.

Un catálogo amplio es sinónimo de variedad de contenidos y un valor añadido para sus dispositivos, pudiendo decantar la elección de un potencial usuario, pero no lo es siempre de calidad. Google Play cuenta con el mayor número de apps pero los controles previos a la publicación son menores que en el App Store de Apple, por lo que se encuentran aplicaciones de baja calidad, con contenido inadecuado o sin adaptar a tabletas. Teniendo en cuenta este aspecto, ambos markets se encuentran prácticamente a la par.

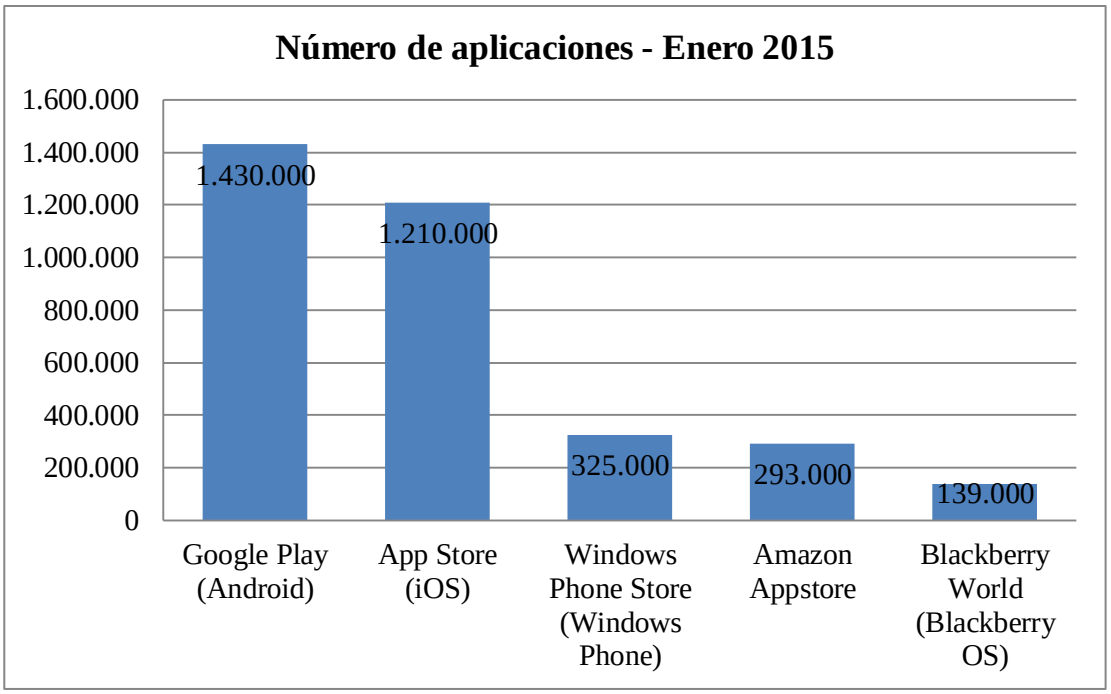


Figura 8. Aplicaciones disponibles en las tiendas digitales de cada plataforma [19]

En cuanto al número de desarrolladores (individuales o compañías) que actualmente tienen publicada una o más aplicaciones, en la Figura 9 se observan los datos extraídos del estudio publicado por el portal appfigures [20] para Google Play, Amazon Appstore y App store. Una vez más, Android se alza con el primer puesto, superando en 21.000 el número de los existentes en iOS.

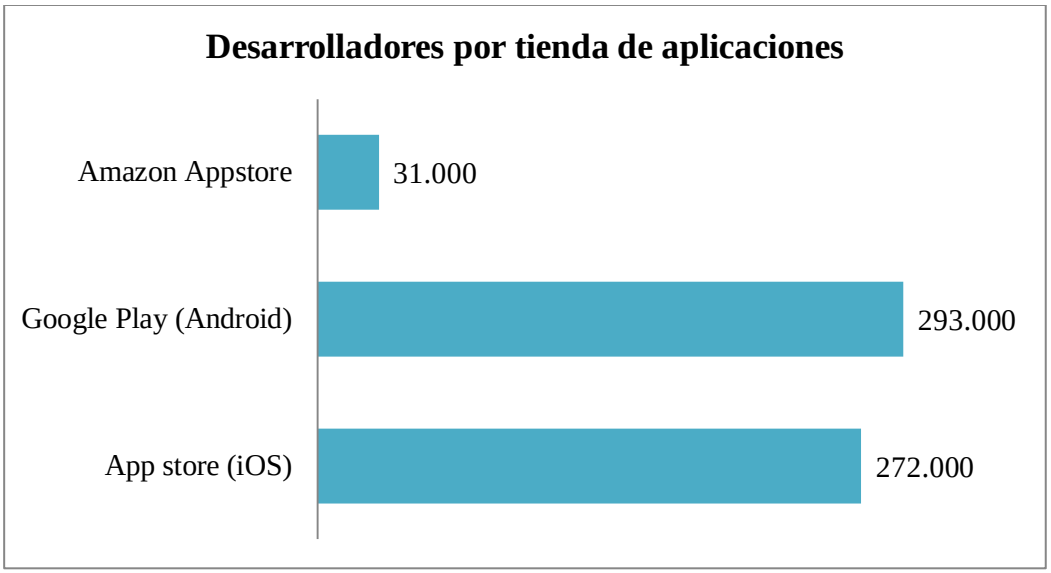


Figura 9. Gráfica que representa el número de desarrolladores activos para las principales tiendas de aplicaciones móviles

Costes de desarrollo y publicación

Cada plataforma cuenta con sus propios IDEs (Integrated Development Environment) y/o SDKs (Software Development Kit), pero dependiendo de la política de la compañía pueden distribuirse gratuitamente o tener algún coste asociado, ya sea único o anual. Posteriormente, una vez desarrollada la aplicación, el proceso de publicación en la tienda digital correspondiente puede ser gratuito o necesitar de un pago extra.

En la Tabla 6 se observa a modo de resumen los costes asociados en el desarrollo para cada uno de los principales sistemas operativos móviles. Blackberry OS supone la opción más económica de partida, al no tener ningún tipo de coste asociado, seguida por Android con un único pago de 25 dólares que otorga derecho de publicación ilimitada en Google Play. Windows Phone OS requiere de un pago anual de 19 dólares para poder publicar en su store, subiendo a 99 dólares anuales en el caso de tratarse de una licencia para empresa. Cabe destacar que los estudiantes disponen de forma gratuita un ticket anual para publicar sus apps. Por último, iOS supone la mayor inversión económica para el desarrollador, ya que al coste anual de 99$ para acceder a las herramientas de desarrollo y App Store, se suma el coste de disponer un dispositivo MAC en el que instalar el entorno de desarrollo.

Sistema operativo	Coste desarrollo	Coste publicación
Android	Gratuito	25$ pago único
iOS	99$ anual	Incluido en coste desarrollo
Windows Phone OS	Gratuito	19$ anual, desarrollador individual 99$ anual, licencia empresa
BlackBerry OS	Gratuito	Gratuito

Tabla 6. Costes asociados al desarrollo y publicación en las plataformas móviles

2.2.1 Conclusión

A la vista de todos los factores anteriormente expuestos, dos plataformas son las que destacan del resto, Android e iOS. La elección final para el desarrollo de la aplicación de este PFC ha sido **Android**, que además de contar con la mayor cuota de mercado, mayor número de aplicaciones, inferior coste de dispositivos y mejores expectativas de futuro, aporta otra serie de ventajas:

- Desarrollo en **Java**, lenguaje muy extendido en la comunidad de desarrolladores, facilitando el acceso a manuales y soporte gratuito y desinteresado.
- Disponibilidad de **SDK multiplataforma** (Windows, Linux y MAC). Supera la barrera impuesta por Apple para el desarrollo en iOS exclusivamente a través de terminales MAC.
- Amplio **número de terminales**. Multitud de marcas fabrican dispositivos con Android, permitiendo elegir uno acorde a nuestras necesidades y/o presupuesto que nos permita testear las aplicaciones desarrolladas en un dispositivo real. El desarrollo en iOS obliga a disponer un teléfono de la gama iPhone, con un catálogo reducido y un coste de adquisición mucho mayor.

2.3 Sistemas de diálogo

2.3.1 Introducción

También conocidos como sistemas conversacionales o SLS (*Spoken Language Systems*), los sistemas de diálogo son una tecnología surgida con el propósito de simplificar la interacción entre persona y ordenador a través del habla natural [21]. Estos sistemas informáticos reciben y generan a su vez frases en lenguaje natural de forma oral, cuya finalidad reside en acercarse lo máximo posible al comportamiento de un ser humano en una conversación con otra persona [22].

Debido a la dificultad que supone desarrollar un sistema como el descrito, suelen orientarse a ámbitos muy concretos con dominios semánticos acotados. Como denominador común a los SLS, existen una serie de operaciones fundamentales que se repiten con cada interacción del usuario [23]:

- *Reconocer* las palabras pronunciadas por el usuario
- Obtener el *significado* de esas palabras para decidir qué información es valiosa en el contexto del sistema.
- *Acceder a base de datos* para obtener la información a trasmitir al usuario o guardar en ella datos de entrada que ha generado.
- Elegir la *respuesta* del sistema al usuario una vez consensuada la acción a realizar en función de la operación solicitada.
- Generar un mensaje *hablado* con la respuesta anterior.

2.3.2 Módulos que conforman un sistema de diálogo

Para poder llevar a cabo el propósito descrito en el apartado anterior, los sistemas de diálogo implementan una arquitectura modular como la mostrada en la Figura 10 [24].

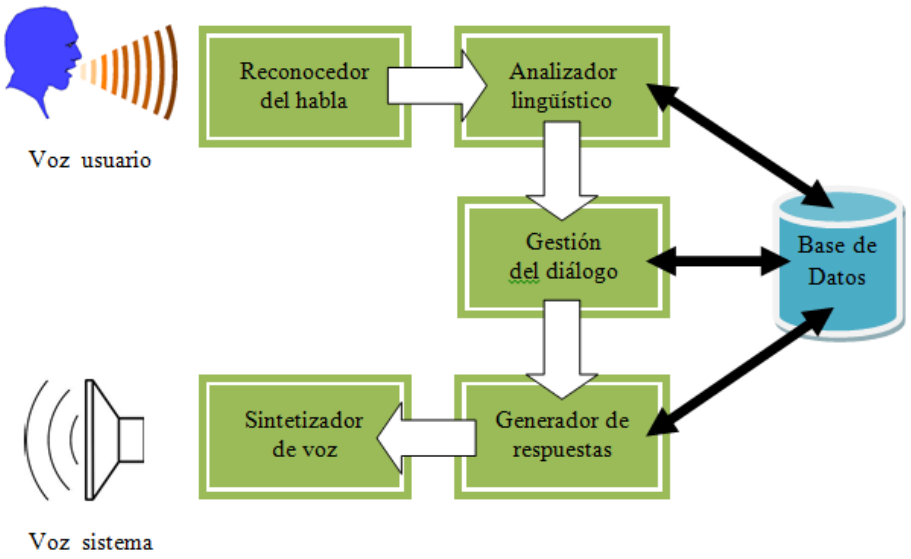


Figura 10. Módulos que conforman la arquitectura de un sistema de diálogo

Módulo de reconocimiento automático del habla

El módulo reconocedor automático del habla (RAH) se encarga de captar y procesar la voz del usuario para su posterior procesado en una cadena de texto, que supondrá la entrada del analizador lingüístico.

Una de las principales barreras a superar por el módulo de reconocimiento de voz es la adecuación a todo tipo de público, incluso al que no ha utilizado nunca un sistema de diálogo. Por ello, a la hora de diseñarlo hay que tener en cuenta una serie de potenciales problemas [25]:

- **Habla continua**. Pueden encontrarse largas secuencias de palabras pronunciadas de forma correlativa, sin pausas marcadas.
- **Independencia del locutor**. Deben adaptarse a las diferencias de pronunciación de los usuarios.
- **Habla espontánea.** Es el factor que más incrementa la tasa de error en el reconocimiento de voz, por la presencia de repeticiones, pausas sonoras, autocorrecciones, etc.
- **Entorno adverso.** Factores como ámbitos ruidosos o micrófonos de baja calidad, suponen entorpecer la tarea del reconocedor de un sistema de diálogo.
- **Lengua.** La diversidad léxica y fonética, incluso sintáctica, es directamente proporcional al número de hablantes de una lengua. A mayor variabilidad es necesaria mayor investigación.

Módulo analizador semántico

También conocido como módulo de procesamiento del lenguaje natural (PLN), su propósito es obtener el significado de las palabras captadas en el módulo reconocedor, ambos módulos forman la entrada del sistema y nutren de información al módulo de gestión del diálogo.

Para cumplir su cometido, el PLN ejecuta un cuádruple análisis [26]:

- *Morfológico*. Decide el género, número y flexión correcta a partir de la raíz de las palabras regulares.
- *Sintáctico*. Establece la estructura gramatical correcta a partir de las unidades léxicas de la oración.
- *Semántico*. Basado en la representación del significado.
- *Pragmático*. Contextualiza la frase analizada.

Módulo gestión del diálogo

El gestor de diálogo (GD) se considera el módulo más importante del SLS, ya que decide cuál es la acción a realizar (generalmente consultar información) cada vez que el RAH procesa la voz del usuario. Un buen diseño de este módulo hará la experiencia de usuario más natural e inteligente al averiguar la información exacta que desea obtener el usuario. Para lograrlo se recurre a recursos como confirmaciones de la información captada al usuario, emprender diálogos que centren la interacción en el dominio del sistema, etc., recuperados en tiempo real de las bases de datos previamente generadas.

Existen tres puntos esenciales en la estrategia de diseñar el GD [23]:

- Grado de **iniciativa**. Para el sistema, el usuario y mixto.
- Forma **confirmación** al usuario. Ya sea explícita, implícita o mixta.
- Medio para la detección y corrección de **errores**. Por ejemplo, cuantificando la confianza de los procesos de reconocimiento y comprensión.

Módulo gestor de base de datos

El módulo gestor de las bases de datos se encarga de gestionar las consultas a las bases de datos existentes y servir al GD los datos recopilados, otorgándole así poder de decisión para afrontar el diálogo.

Módulo generador de respuestas

El módulo generador de lenguaje natural (GLN) se nutre de la información obtenida del GD para componer una respuesta gramatical y semánticamente correcta en formato texto, a partir de un grupo de patrones internos del sistema. Las frases generadas como respuesta deberán ser lo más fiel posible al lenguaje natural.

La interfaz de salida de los sistemas de diálogo la componen el tándem GLN y sintetizador de voz.

Módulo sintetizador de voz

Para finalizar, el módulo de síntesis de voz o TTS (acrónimo inglés de *Text-to-Speech System*) se encarga de transformar a formato sonoro la respuesta en texto generada por el sistema. A nivel funcional cumple una doble función, por una parte proporcionar una respuesta verbal a las indicaciones o demandas del usuario y por otro lado parafrasearlas a modo de confirmación (previamente generadas por el GD).

La estructura del sintetizador responde a la necesidad de dotar al sistema de información lingüística apropiada para transformar los caracteres ortográficos en señal acústica [21]:

- *Procesamiento del texto*. Se descartan signos de puntuación sin valor lingüístico, símbolos, números, abreviaturas y siglas.
- *Análisis lingüístico*. Es muy importante para elegir el acento en palabras homógrafas, establecer las pausas no marcadas, o en definitiva cualquier circunstancia especial susceptible de análisis.
- *Transcripción fonética.* Refleja la conversión en pronunciación a partir de caracteres fonéticos.
- *Asignación elementos prosódicos*. Como son la duración e intensidad de los sonidos, las pausas o la entonación. Primordial para una aproximación al lenguaje natural.
- *Diccionario de unidades de síntesis.* Mediante la concatenación de un número limitado de fonemas es posible generar cualquier texto de una lengua concreta.
- *Conversión en valores acústicos.* Es la generación en onda acústica aplicando de nuevo ajustes prosódicos para obtener una entonación correcta.

2.3.3 Sistemas de diálogo multimodal

El creciente interés por los SLS ha promovido una gran investigación en el campo, evolucionando y perfeccionando cada uno de sus módulos, así como solventando ciertas limitaciones detectadas. Quizás la mayor de ellas sea la comunicación exclusiva por medio del habla, al limitar la información de entrada-salida al sistema. Para solventar esta problemática, los **sistemas de diálogo multimodal o MDS** (acrónimo de *multimodal dialogue system*) incorporan otros medios de entrada como una pantalla táctil, un teclado, un ratón, etc. De igual forma, el canal de salida posee múltiples opciones como texto, voz, imágenes, o cualquier elemento que estimule los sentidos del usuario [22].

Existen además sistemas multimodales que posibilitan la elección del método de entrada y/o salida al usuario, con el fin de lograr una adaptabilidad en función a las capacidades físicas del individuo, como sucede con personas invidentes, sordomudas, ancianas, o cualquier otro colectivo que pueda encontrar barreras de acceso a los sistemas informáticos. También permiten adaptarse al entorno, condiciones ambientales adversas como la falta o el exceso de iluminación, lugares ruidosos, etc., pueden verse paliadas gracias a la elección del método de entrada/salida más acorde para esa circunstancia. Con este método se maximiza la cantidad de información que un usuario concreto puede aportar a un sistema determinado y viceversa.

Al permitir añadir una componente visual, los MDS se aproximan más a la comunicación humana, basada en gestos, expresiones faciales y miradas además de la voz [24].

2.3.4 Aplicaciones de los sistemas de diálogo

Existen multitud de aplicaciones, tanto comerciales como de ámbito investigador, basadas en sistemas de diálogo (hablado o multimodal) para automatizar el acceso a la información. A continuación se listan a modo de ejemplo algunas de ellas, categorizadas por la función que desarrollan:

❖ Información y reservas en comercio electrónico

Voice Ticketing [27]

Servicio que permite automatizar los procesos de compra-venta de entradas o tickets de distintos servicios. Mediante el uso del reconocimiento de voz y de la síntesis de texto a voz, facilita el filtrado y búsqueda de información.

Natural Vox [28]

Empresa especializada en SLS, ofrece automatismos en procesos telefónicos. Su abanico de productos permite realizar operaciones bancarias, pedir citas previas, servir información sobre la bolsa, climatología, control de pacientes crónicos mediante llamadas rutinarias, gestión de seguros, etc.

❖ Información meteorológica

ATTEMPS [29]

Desarrollado por la Universidad Politécnica de Cataluña, se trata de un sistema de diálogo multimodal que proporciona información meteorológica mediante comunicación telefónica. Indicando una localidad dentro de Cataluña, el sistema proporciona un parte meteorológico detallado. La interacción multimodal se obtiene mediante el sistema de alertas (establecidas por el Servicio Meteorológico de Cataluña) y avisos (establecidos por el usuario mediante llamada telefónica), que mediante texto (SMS) o voz (llamada telefónica) es capaz de llevar al usuario el mensaje designado en el sistema.

❖ Fines educativos y terapéuticos

VOCALIZA [30]

Diseñado por el Communicatins Technology Group (GTC) de la Universidad de Zaragoza, se trata de un sistema multimodal utilizado en la terapia de ciertas patologías del habla en castellano. Mediante un sistema que evalúa la calidad de la pronunciación, el usuario es debe repetir el ejercicio presentado mediante refuerzo visual hasta que su pronunciación sea aceptada. Existe un grupo de ejercicios diseñados para entrenar factores fonológicos, sintácticos y semánticos, mediante la repetición de palabras, frases y adivinanzas simples. En la Figura 11 puede observarse un ejemplo de ejercicio basado en la repetición de la palabra “ÁRBOL”.

Figura 11. Pantalla de entrenamiento de VOCALIZA

Para lograr su objetivo la aplicación se basa en cuatro tecnologías: un reconocedor automático del habla, un sintetizador de texto a voz, un sistema de adaptación al usuario y por último un módulo de verificación del habla.

❖ Control de entornos inteligentes

MIMUS [31]

MIMUS es un sistema de diálogo multimodal y multilenguaje desarrollado para controlar un hogar inteligente, en el que ciertos elementos se encuentran automatizados. Se diseñó pensando originalmente en facilitar ciertas actividades cotidianas a personas discapacitadas. La interfaz con la que realiza la comunicación el usuario es protagonizada por un agente virtual que simula una persona. El sistema es compatible con los idiomas inglés, alemán y español.

❖ Agentes virtuales

Siri

Es el asistente personal que Apple incluye en sus dispositivos de la plataforma iOS desde 2010. Gracias a este sistema podemos controlar múltiples funciones del terminal gracias a indicaciones de voz en lenguaje natural, utilizando el interfaz gráfico mostrado en la Figura 12. Las principales tareas que suceden dentro del sistema son [32]:

- Reconocimiento automático de la voz para transcribir las órdenes concretas o información a buscar en texto.
- Procesado del lenguaje natural del texto obtenido por el ASR, para obtener un texto correctamente formateado.

- Análisis de preguntas y órdenes del usuario en el texto procesado.
- Mediante un servicio web de terceros, se realiza una búsqueda de la información necesaria para mostrar al usuario.
- Transformación a lenguaje natural de la información recopilada y formateada para presentar al usuario.
- Uso de sintetizador de texto a voz para comunicar al usuario el resultado de la operación y/o búsqueda.

Figura 12. Interfaz de Siri, diálogo de bienvenida

Fuente: http://maclovers.universiablogs.net/files/Siri-post.png

Gracias a Siri, el usuario puede gestionar calendarios y alarmas, realizar llamadas, gestionar el correo electrónico, publicar en redes sociales como Facebook y Twitter, buscar amigos gracias a servicios de geolocalización, búsqueda de información general en la red, etc.

❖ Consulta de información

SmartKom [24]

Sistema multimodal desarrollado por el Centro para la Inteligencia Artificial alemán, añade comunicación mediante movimientos de la mano y expresiones faciales a la habitual mediante la voz. La mayor peculiaridad es el mecanismo de proyección para generar gráficos sobre superficies e interactuar con ellos señalizándolos. Adicionalmente, la proyección genera un agente animado con voz y movimientos sincronizados. Dadas sus características es apropiado para un amplio dominio de aplicaciones. En la Figura 13 se observa un ejemplo de uso.

Figura 13. Interfaz proyectada por SmartKom

AUGUST [33] [24]

Sistema de diálogo multimodal desarrollado por el Centre for Speech Technology (CTT), Royal Institute of Technology (KTH) de Suecia y expuesto en el Centro Cultural de Estocolmo. Mediante un agente animado inspirado en el dramaturgo sueco August Strindberg, se emplean la voz y expresiones faciales junto con movimientos de la cabeza para comunicarse con el usuario en varios dominios temáticos, como son información biográfica del escritor y de restaurantes de la ciudad de Estocolmo. El agente es complementado mediante un sistema de bocadillos a modo de viñeta para presentar información adicional (Figura 14).

Figura 14. Imagen del el asistente August junto con el sistema de información mediante bocadillos

AdApt [33]

Sistema de diálogo multimodal también desarrollado por el Centre for Speech Technology (CTT), Royal Institute of Technology (KTH) de Suecia, con el propósito de ayudar al usuario en la tarea de encontrar apartamento en Estocolmo. AdApt pregunta al usuario una serie de datos y el sistema guía en la elección, proporcionando información detallada de apartamentos recuperados en tiempo real de la web y que cumplen las características indicadas. Por lo tanto se centra en un dominio concreto de aplicación.

Además de la iteración mediante voz, existe la posibilidad de pulsar en el icono de un apartamento o marcar áreas del mapa interactivo de Estocolmo. En la Figura 15 puede observarse la interfaz de usuario donde se observa el mapa indicado y el asistente animado.

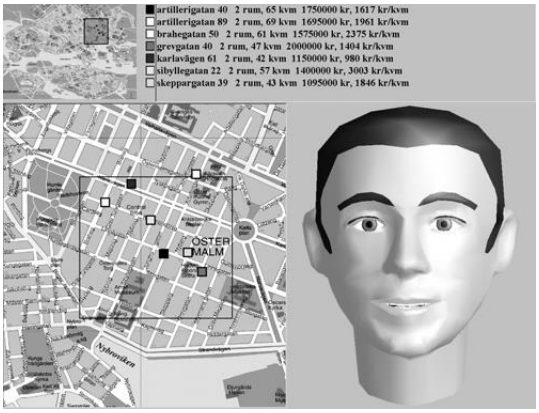


Figura 15. Interfaz de usuario de sistema de diálogo multimodal AdApt

2.4 Sistemas de reconocimiento de voz en Android

Desde los inicios del sistema operativo, Google ha apostado por utilizar métodos de entrada e interacción con el dispositivo alternativos al teclado (físico o virtual), principalmente mediante el reconocimiento de la voz del usuario. A continuación se describen todas las funcionalidades Android que hacen uso de la tecnología ASR.

2.4.1 Búsqueda por voz

Incorporada en Android 1.6, permite realizar búsquedas de información en el buscador de Google por medio del habla. Al pulsar el icono de la aplicación (Figura 16) o la imagen del micrófono a la izquierda de la barra de búsqueda de Google en la pantalla principal, emerge una ventana con el mensaje "Hablar ahora" que indica el momento en el que empieza a recopilar los términos de búsqueda pronunciados por la persona (Figura 17).

Figura 16. Icono de aplicación Búsqueda por voz de Google [34]

Figura 17. Ventana emergente de aplicación Búsqueda por Voz de Google [35]

2.4.2 Dictado por Voz

El dictado por voz, integrado por primera vez en Android 2.1, se refiere al proceso de introducción de texto en las aplicaciones a partir de la voz del usuario. De esa manera, podemos redactar un mensaje de texto o correo, escribir en programas de mensajería instantánea como WhatsApp, etc., sin necesidad de utilizar el teclado táctil, únicamente por medio del habla.

Habilitar el dictado por voz es tan sencillo como acceder a *Ajustes>Idioma e introducción de texto>Teclado y métodos de introducción* y pulsar en *Dictado por voz de Google*. Después es necesario acceder a las propiedades del teclado instalado y activar en preferencias la opción *Tecla de entrada de voz.* Ambas opciones pueden observarse en la Figura 18. El lugar exacto para realizar la configuración puede variar entre versiones de Android, la figura anterior corresponde a la versión 4.4 (KitKat).

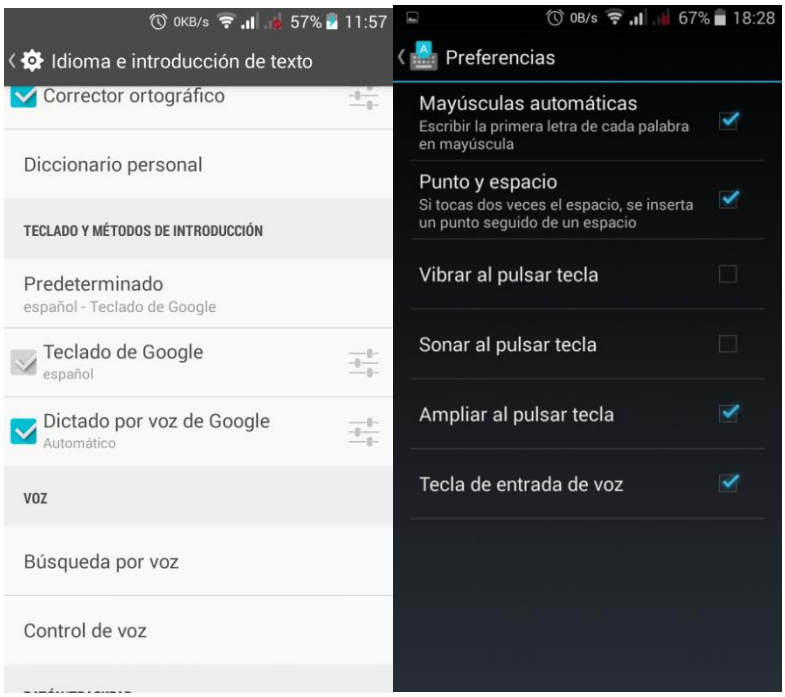


Figura 18. Selección del método de entrada por Dictado de voz de Google en Android 4.4

Para iniciar el dictado a texto por voz, es necesario pulsar en el campo de texto a rellenar para que se muestre el teclado, después basta con pulsar el símbolo del micrófono para iniciar el proceso. Con la evolución de este sistema, actualmente son soportados signos de puntuación en español, es decir, podemos indicar "punto", "coma", "signo de interrogación" o "nueva línea". Al acabar de dictar, la pausa es detectada y el reconocedor de lenguaje transcribirá todo lo expuesto [36]. En la Figuras 19 y 20 se muestran ejemplos reales de uso en una aplicación de mensajería SMS y el gestor de correo electrónico Outlook para Android respectivamente.

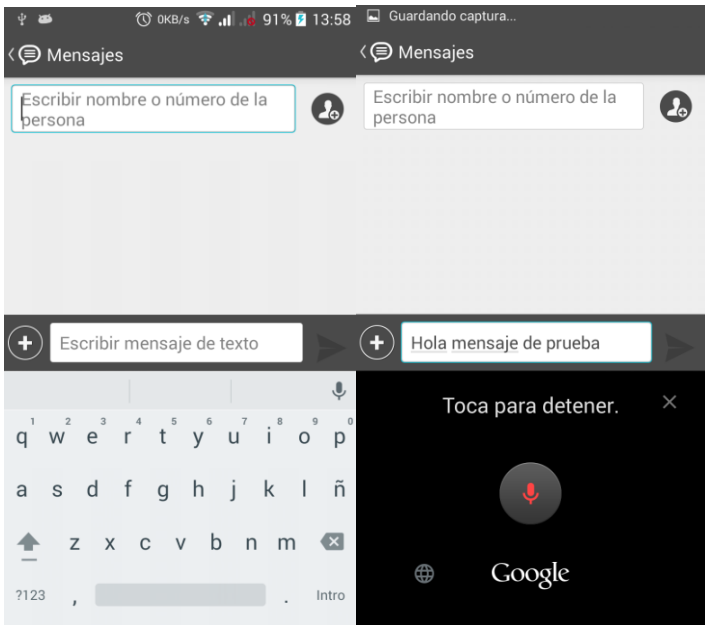


Figura 19. Dictado de voz en redacción de mensaje SMS

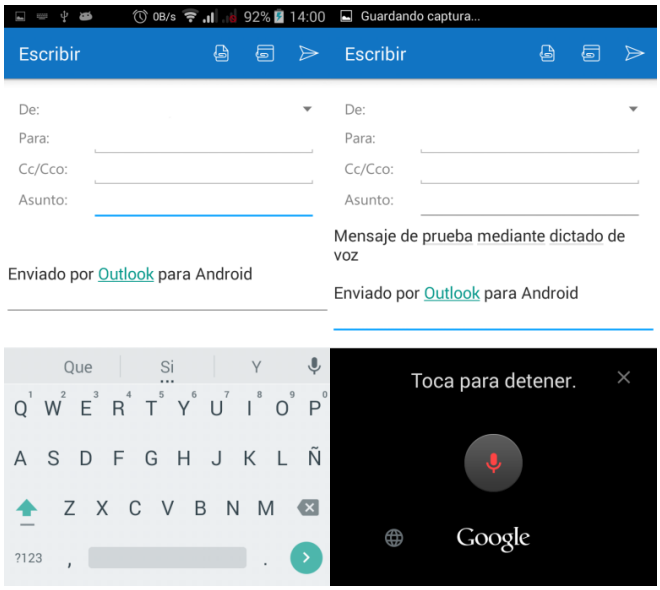


Figura 20. Dictado de voz en redacción de correo electrónico

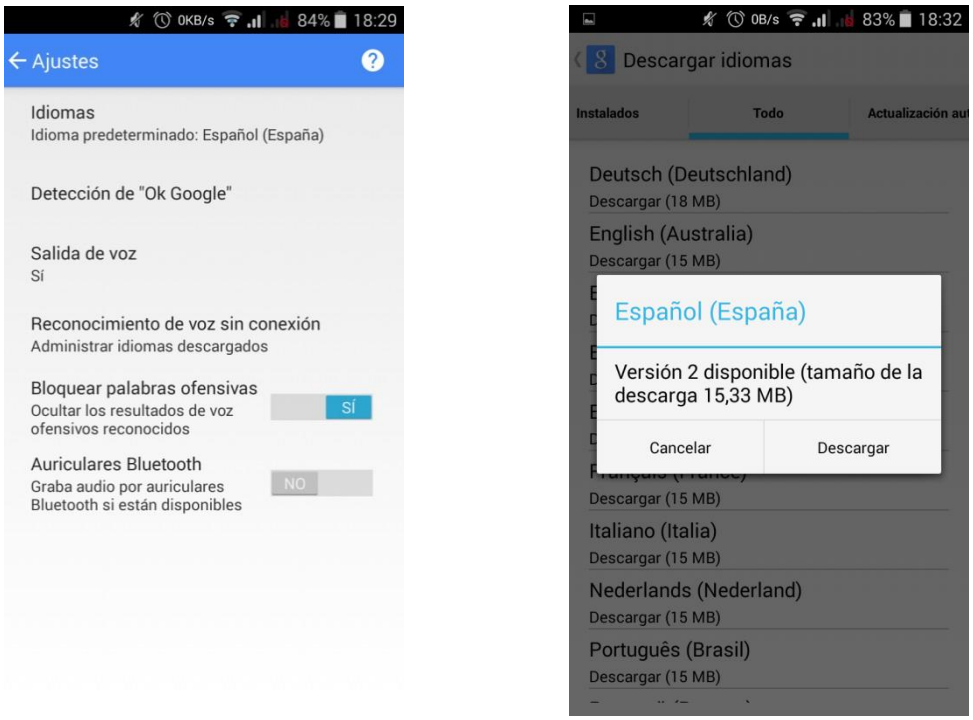
Con Android 4.1 (Jelly Bean) se introduce la posibilidad de usar el **reconocimiento de voz offline**, es decir, sin conexión a Internet. Hasta ese momento, era necesario conectar con los servidores de Google para realizar la interpretación del lenguaje natural emitido por el usuario.

Para habilitarlo, únicamente es necesario realizar los siguientes pasos (Figura 21):

- Asegurarnos de tener la aplicación Google Search actualizada a la última versión y en caso contrario actualizarla. Lo comprobamos desde Google Play.

- En el menú de “Ajustes” del dispositivo, “Idioma e introducción de texto”, “Búsqueda por voz” y “Reconocimiento de voz sin conexión” (Figura 21(a)).
- Seleccionar la pestaña “Todo”, elegir el idioma Español (España) y pulsar en “Descargar” (Figura 21(b)).
- Por último, es necesario tener habilitado en el terminal la salida de texto a voz. Este punto se explica en detalle en el **Apartado 2.4** de esta memoria.

A partir de ese momento, el reconocimiento de voz a texto se efectuará de forma local en el dispositivo.



(a) Acceso a la administración del reconocimiento de voz sin conexión

(b) Descarga del idioma Español en el dispositivo

Figura 21. Activación y elección de idioma del reconocimiento de voz offline

2.4.3 Acciones de voz

En el lanzamiento de la **versión 2.2 de Android** se descubre una nueva funcionalidad de la aplicación Google Search, denominada ***Voice Actions*** [37]. Permite dar instrucciones de voz al teléfono para realizar variedad de tareas a través de las aplicaciones del terminal.

Para utilizar las acciones de voz, de igual manera que en la búsqueda por voz (**Apartado 2.4.1**), basta con pulsar el micrófono de la barra de búsqueda de Google, situado en la pantalla principal. Al mostrarse la ventana *Speak Now/Hable ahora* podrá comenzar la introducción de las instrucciones por medio del habla.

Aunque en la presentación del servicio sólo se encontraba disponible para Estados Unidos en idioma inglés, un año después se expande a algunos países europeos soportando su idioma local, entre los que se encuentra España [38].

En la Tabla 7 se muestran todos los comandos de voz soportados en el idioma español junto con la aplicación del sistema implicada en la operación y un ejemplo práctico de uso.

Comando de voz	**Aplicación implicada**	**Ejemplo de uso**
Enviar mensaje a [contacto] [mensaje]	Mensajería SMS.	"Enviar mensaje a Pablo. Hola, nos vemos en la cafetería en dos horas"
Llamar a [nombre de la empresa]	Marcador telefónico o *dialer*.	"Llamar a floristería La petunia"
Llamar a [contacto]	Marcador telefónico o *dialer*.	"Llamar a Luis"
Ir a [página web]	Navegador web.	"Ir a forocoches"
Cómo llegar a [dirección/nombre comercio]	Google Maps.	"Cómo llegar a calle Serrano 58 Madrid"
Direcciones para llegar a [lugar/nombre comercio]	Google Maps.	"Direcciones para llegar a Museo Nacional del Prado"
Mapa de [lugar]	Google Maps.	"Mapa de Leganés"

Tabla 7. Acciones de voz disponibles para idioma español en Android 2.2

La versión inglesa dispone de tres acciones adicionales que no llegaron a España en Android 2.2, ***listen to*** *[artista/song/album]* para escuchar música con el criterio de búsqueda indicado a través de alguna aplicación musical del dispositivo como por ejemplo Pandora, Last.fm, Rdio, mSpot; ***send email to*** *[contact][message]* para envío de correo electrónico a través del gestor predeterminado y ***note to self*** *[note]*para guardar una nota [37].

Con **Android 4.1** se presenta **Google Now** como el asistente personal inteligente de Google en respuesta a Siri de Apple. Supone una evolución tanto de Google Search (búsquedas de texto) como de Voice Search (búsquedas por voz) al introducir capacidad de reconocimiento de lenguaje natural en ambos sistemas, gracias al proyecto con nombre en código "**Majel**". En paralelo, gracias al proyecto "**Knowledge Graph",** se monitoriza cierta información del usuario como el historial de búsquedas en Google, su geolocalización, contenido de Gmail, calendarios, etc., para poder representar sugerencias de información y contenido en forma de tarjetas dentro de la propia interfaz de Google Now [39]. En la Figura 22 se observa la interfaz de Google Now, con sus fichas y doodles personalizados.

Figura 22. Interfaz de la pantalla de inicio de Google Now

En relación a la evolución de las acciones de voz en Android 4.1, existe un gran progreso gracias al proyecto *Majel* anteriormente nombrado pero también gracias a la función **Ok Google**. Basta pronunciar esas palabras desde la interfaz de Google Now o de la Búsqueda de Google para comenzar el proceso de escucha de comandos de voz o términos de búsqueda. Por ejemplo, puede pronunciarse "Ok Google, mapa de Madrid" para obtener un resultado de la búsqueda complementado con información de voz por parte del buscador: "Éste es el mapa de la zona cercana a Madrid" (Figura 23).

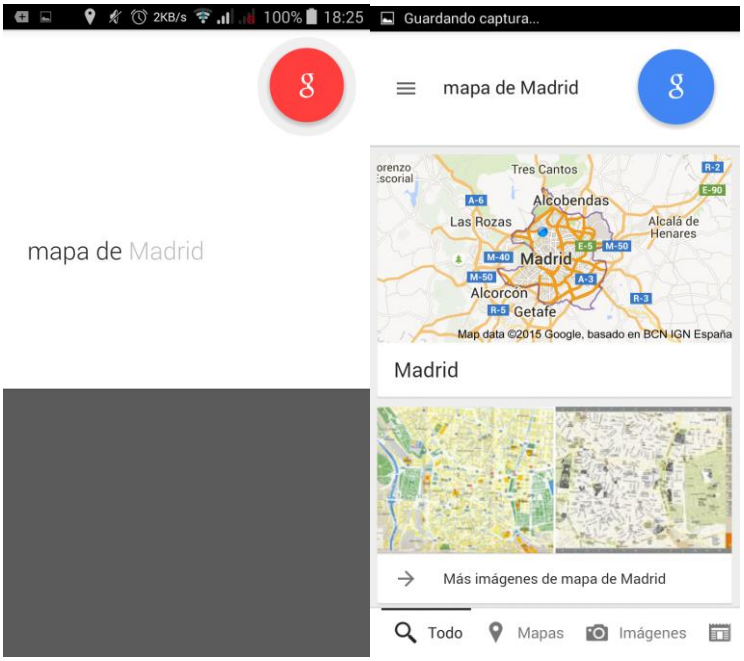


Figura 23. Ejemplo de uso de Ok Google solicitando un mapa de Madrid

Google ha decidido ir un paso más allá e integrar Google Search/Voice Search con las aplicaciones del dispositivo que hayan sido adaptadas para tal propósito, como indican desde el blog de Android Developers [40]. De esta manera se pueden realizar consultas del tipo *"Hoteles cercanos en Tripadvisor"*.

En sus orígenes, Ok Google podía lanzarse desde la interfaz de Google Now o en caso de no encontrarse habilitado, desde Google Search. Posteriormente, con el Google Now Launcher desarrollado por Google las palabras podían ser pronunciadas desde el escritorio

Para activar Ok Google en español, se requiere al menos la versión 3.15 de Google Search [41] y acceder a *Ajustes>Idioma e introducción de texto>Voz>Búsqueda por voz* y pulsar en *Detección de "Ok Google"*, estableciendo a *Sí* la opción. En la Figura 24 se observa gráficamente estas opciones de configuración.

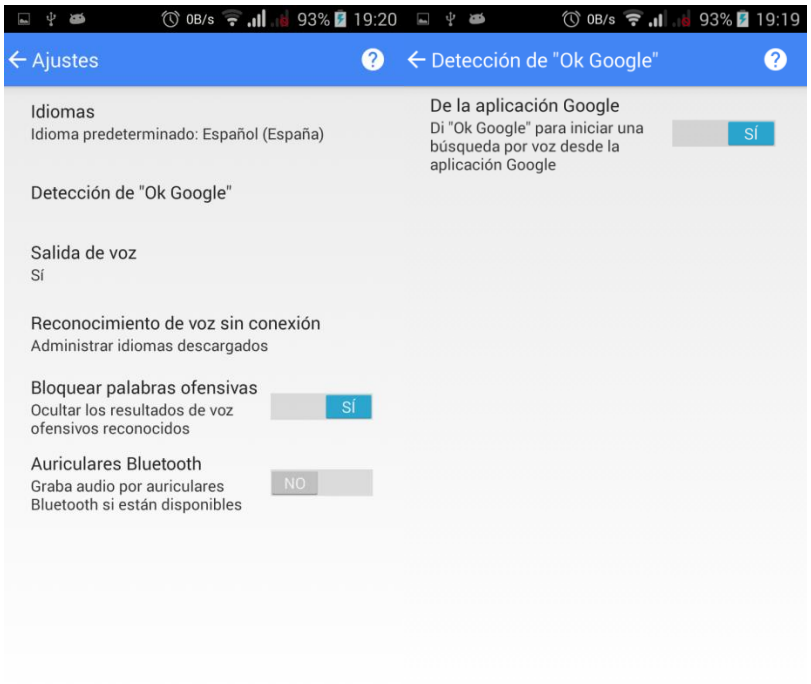


Figura 24. Proceso de activación de Ok Google

La lista de comandos de voz soportados por *Ok Google* desde Android 4.1 es netamente mayor a la disponible en las *Voice Actions* de la versión 2.2, y sigue creciendo constantemente. En la Tabla 8 [42] se muestran los comandos más significativos disponibles hasta la fecha. Algunos de estos comandos se ilustran en la Figura 25.

Temática	Comandos de voz
Tiempo	▪ ¿Qué temperatura va a hacer el [día] [por la mañana / por la tarde /por la noche]?
	▪ ¿Va a llover mañana o va a llover el [día]?
	▪ ¿Tiempo en [país o ciudad]?
Notas y recordatorios	▪ Recuérdame [acción o tarea] en [cantidad de horas] – *Ejemplo: recuérdame comprar comida en 2 horas*
	▪ Despiértame en [cantidad de horas] – *Ejemplo: despiértame en 8 horas*
	▪ Establecer alarma a las [hora] – *Ejemplo: Establecer alarma a las 6 de la tarde*
	▪ Añadir nota [texto]
Música	▪ Escuchar [nombre canción/nombre artista] en [Play Music/Youtube]
	▪ Canciones [artista]
	▪ Álbumes [artista]
	▪ Compañía, miembros o grupos [artistas o grupo]
	▪ Obras [artista]
Información horaria	▪ Hora en [país]
	▪ Cuando [amanece/anochece] en [país]
	▪ Zona horaria en [ciudad/país]
Calculadora	▪ Cuantos [tipo moneda] son 1000 [tipo moneda]
	▪ [Número] por ciento de [cifra] – *Ejemplo:48 por ciento de 2000*
	▪ Convertir [cifra] kilómetros en millas.
	▪ Raíz cuadrada de [número]
	▪ [Número] [operador] [número][operador][número]… - *Ejemplo: 6 por 5 dividido por 3*
Cine	▪ ¿Reparto de [serie / película]?
	▪ ¿Quién produjo [película]?
	▪ ¿Cuándo se estrenó [película]?
	▪ ¿Cuánto dura la película [título]?
	▪ ¿Qué películas hizo [director]?
	▪ ¿En qué películas ha actuado [actor/actriz]?
	▪ Fecha de estreno, primer episodio, director, reparto, música, creador, escritores o premios [película/serie]

	▪ Películas, nominaciones o premios [actor/actriz]
Deporte	▪ ¿Cuándo juega [equipo deportivo]?
	▪ Calendario de [equipo deportivo]
	▪ Resultado de [equipo deportivo]
	▪ Salario de [deportista]
	▪ Equipos de [deportista]
	▪ Premios [deportista]
	▪ Capacidad de [estadio deportivo]
	▪ Inauguración de [estadio deportivo]
	▪ Cierre de [estadio deportivo]
	▪ Arquitecto de [estadio deportivo]
Geografía	▪ Población, superficie, lugares de interés [lugar/ciudad]
	▪ Superficie, profundidad o islas [Mar/océano]
	▪ Capital de [país]
	▪ Donde está el edificio más alto del mundo
	▪ Edad de la tierra
	▪ Longitud, caudal, boca, puentes, ciudades o países [río]
General	▪ ¿Quién fundó [compañía]?
	▪ ¿Quién es el CEO de la + (compañía)?
	▪ Conocer la estatura, peso, edad, fecha de nacimiento, educación, fecha de muerte, causa de muerte, lugar de nacimiento, pareja, padres, hermanos, hijos + (personaje)
	▪ Altura, plantas, inauguración, función, estilo arquitectónico o arquitecto + (monumento o edificio)
	▪ Definición de + (palabra)
	▪ ¿Quién escribió + (título libro)?
	▪ ¿Cuándo [nació/murió] [personaje]?
	▪ Inicio de construcción + (monumento o edificio)
	▪ ¿Cuáles son las dimensiones de + (obra de arte)?
	▪ ¿Quién inventó [objeto]?

	▪ ¿Quién es [personaje]?
Imágenes	▪ Imágenes de [monumento] [al atardecer/al amanecer/al anochecer]
	▪ Muéstrame imágenes de [nombre][acción] – *Ejemplo: muéstrame imágenes de Batman volando*
	▪ Logo de [empresa]
Mapas y navegación	▪ Mapa de [municipio/ciudad/país/continente]
	▪ Llévame a [negocio/comercio][a pie/en coche]
	▪ ¿Dónde está el [museo/monumento] de [ciudad]?
	▪ ¿Dónde está [localidad/municipio/colegio]?
	▪ [local de comida][ciudad]
	▪ Llamar al hotel [nombre][ciudad]
	▪ ¿Cómo llegar a [nombre negocio] [ciudad]?
	▪ Distancia de [lugar] a [otro lugar] – *Ejemplo: distancia entre Madrid y Toledo*
Internet	▪ Ir a /abrir / muéstrame [URL]
Aplicaciones	▪ Abrir [nombre aplicación]– *Ejemplo: Abrir Twitter*
Llamadas y mensajes	▪ Llamar a [contacto]
	▪ Enviar mensaje a [contacto] mensaje [texto]

Tabla 8. Principales comandos de voz para Ok Google

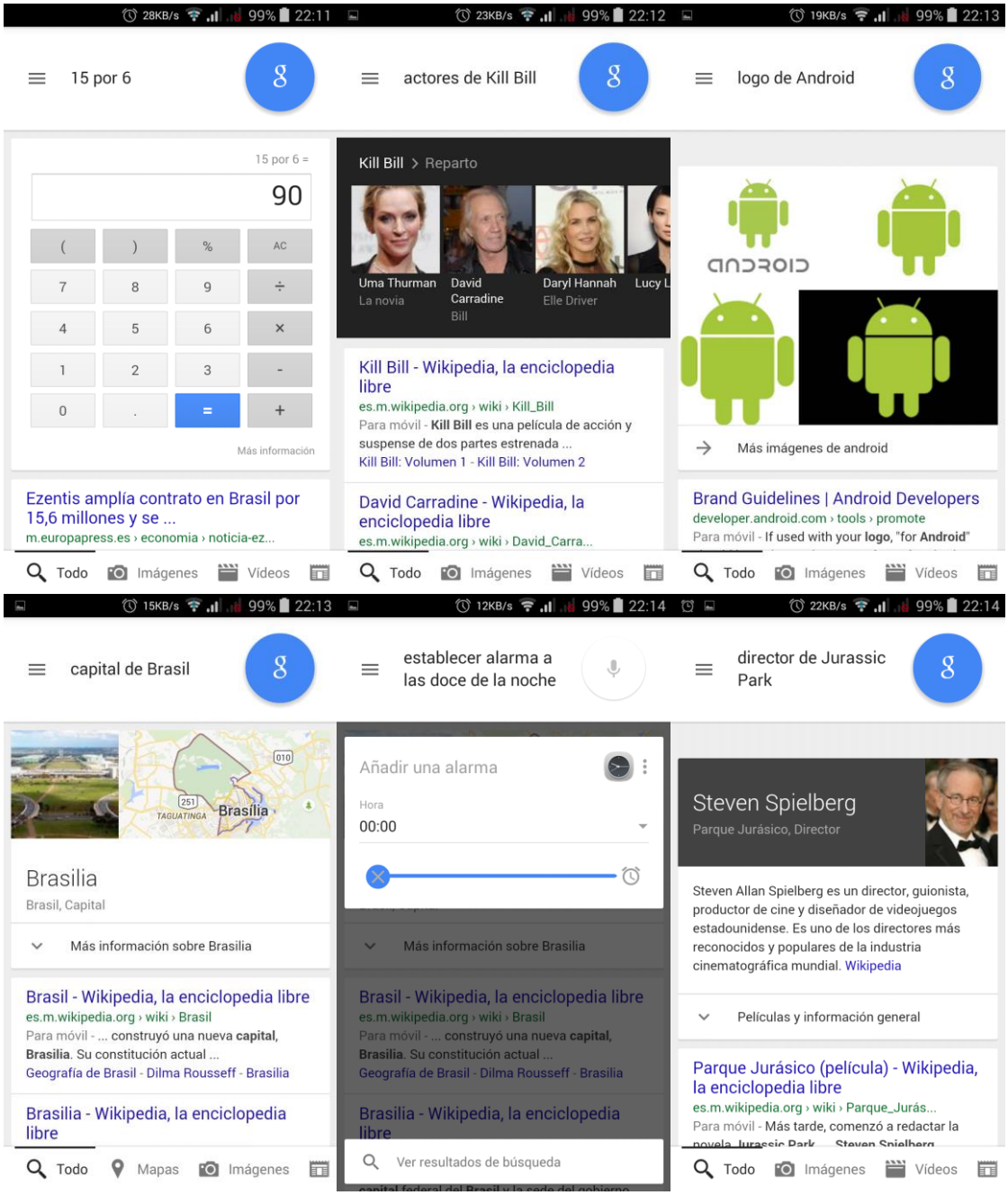


Figura 25. Ejemplos de uso de Ok Google en Android 4.1

2.4.4 Librerías Android ASR

Mediante el SDK de Android, se pone a disposición de los desarrolladores la posibilidad de introducir reconocimiento automático del habla para enriquecer la interacción con aplicaciones de terceros. Desde el paquete ***android.speech*** introducido en el nivel de API 3 se definen una interfaz y un conjunto de clases para integrar y adaptar el sistema ASR a las necesidades del desarrollador. En la Tabla 9 se muestran las descripciones de cada uno de los componentes del paquete *android.speech* [43].

Tipo	Componente	Descripción
Interfaz	RecognitionListener	Recibe las notificaciones lanzadas por la clase *SpeechRecognizer* cuando sucede un evento de reconocimiento.
Clase	RecognitionService	Provee la base para implementar el servicio de reconocimiento. Sólo debe ser extendida en caso de implementar un nuevo reconocedor de voz.
	RecognitionService.Callback	Clase que recoge los resultados del servicio de reconocimiento de voz y los reenvía al usuario. Una instancia de esta clase es enviada al *RecognitionService* a través del método *onStartListening(Intent, Callback)*. Los métodos de esta clase pueden ser invocados desde cualquier hilo de ejecución.
	RecognizerIntent	Contiene las constantes que definen las preferencias y son distribuidas al reconocedor de voz por medio de un *intent*.
	RecognizerResultsIntent	Contiene las constantes de los intents implicados en mostrar los resultados del reconocimiento de voz.
	SpeechRecognizer	Gestiona la interacción con el servicio reconocedor del habla, realizada en los servidores de Google. No se debe instanciar esta clase directamente, en su lugar debe utilizarse el constructor *createSpeechRecognier(Context)*. Los métodos de esta clase sólo deben ser invocados desde el hilo principal de la aplicación. Se requiere el permiso de aplicación RECORD_AUDIO.

Tabla 9. Componentes del paquete android.speech

2.5 Sistemas de texto a voz en Android

Desde Android 1.6 se incorpora por defecto un motor TTS para permitir "hablar" a las aplicaciones [44]. En las primeras versiones se trata de *Pico TTS* desarrollado por SVOX pero se ha visto sustituido por la *Síntesis de Google* en versiones más recientes. Aunque no es frecuente, existen dispositivos sin ningún tipo de motor TTS preinstalado.

Para habilitar la síntesis de texto a voz en el sistema, debe existir previamente o proceder a instalar en caso contrario, al menos una aplicación que ejerza de motor de conversión. Acto seguido desde *Ajustes>Salida de texto a voz* se selecciona cualquiera de los motores disponibles (Figura 26) y en los ajustes del motor se elige el idioma español y se habilitan las opciones correspondientes a actualización automática de voces y uso único de Wi-Fi, para evitar consumo de datos asociados a tarifa de proveedor de red (Figura 27).

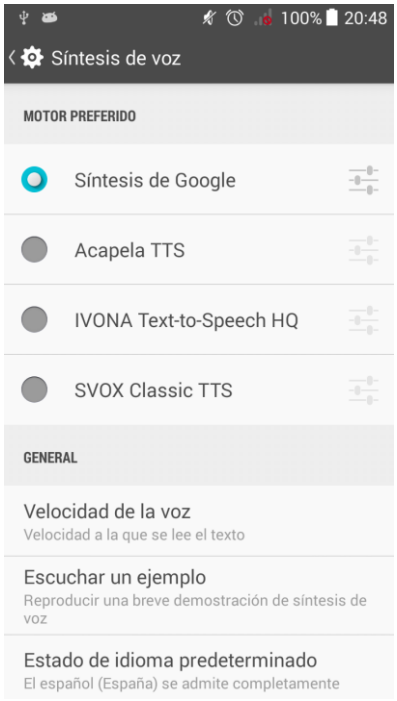


Figura 26. Pantalla de selección de motor de síntesis de voz y opciones disponibles

A continuación pueden realizarse una serie de acciones ubicadas debajo de la lista de motores disponibles. Mediante *Velocidad de la voz* se puede seleccionar un valor desde "Muy lenta" a "Lo más rápido posible". Gracias a *Escuchar un ejemplo* se puede verificar el correcto funcionamiento así como las características sonoras de la voz elegida, como pueden ser naturalidad del lenguaje, velocidad, presencia de voz robotizada, etc.

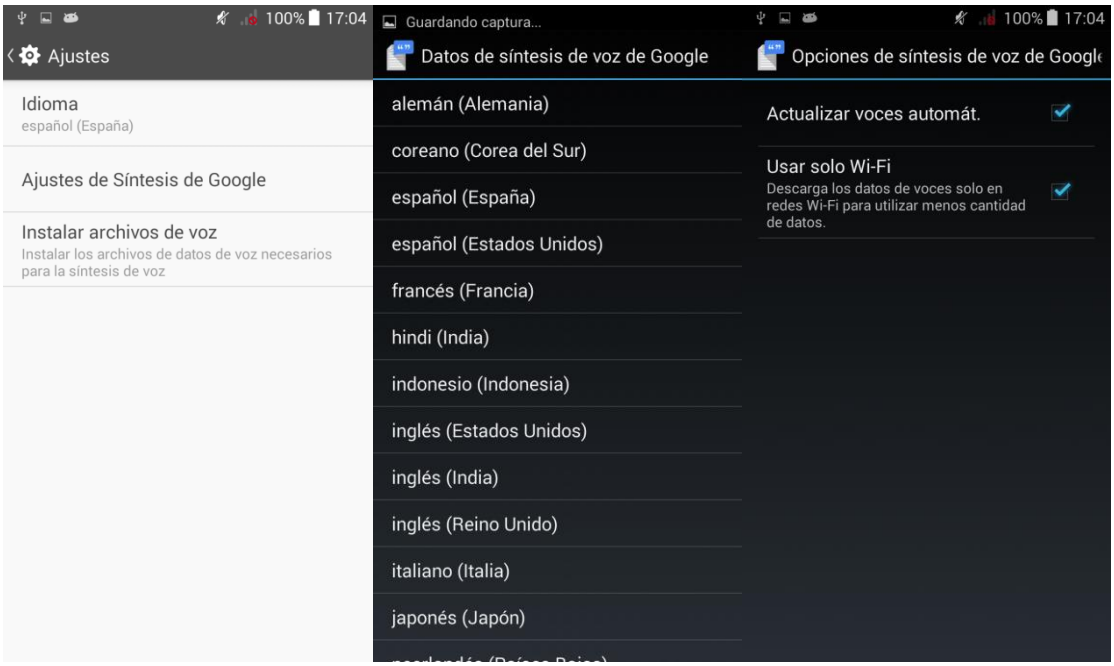


Figura 27. Ajustes de motor Síntesis de Google, selección y descarga de idiomas

2.5.1 Motores de síntesis de texto a voz

La Tabla 10 muestra ciertas características de los principales motores TTS con lenguaje español que se pueden encontrar actualmente en la tienda de aplicaciones Google Play. A continuación se muestran cada una de las características analizadas:

- **Versión**. Refleja la versión de la aplicación analizada.
- **Desarrollador**. Compañía o desarrollador responsable del motor de síntesis de voz.
- **Idiomas**. Número total de voces soportadas. Se representan por un idioma y sexo (hombre o mujer).
- **Realismo**. Cuantifica de manera subjetiva el realismo percibido de las voces emitidas por el motor TTS.
- **Calidad audio**. Distingue el nivel de calidad del audio reproducido entre estándar y alta calidad.
- **Ubicación**. Es la URL a la aplicación en Google Play.
- **Compatibilidad.** Indica las versiones de Android soportadas por el motor de síntesis analizado.
- **Tamaño.** Cuantifica el tamaño de la aplicación instalada y de las voces descargadas.
- **Precio.** Muestra el precio de la aplicación y/o voces disponibles.

Motor TTS	Versión	Desarrollador	Idiomas	Realismo	Calidad audio	Ubicación	Compatibilidad	Tamaño	Precio
Síntesis de voz de Google	3.4.6.1819666	Google	16 - Alemán, coreano, español (España), español (Estados Unidos), francés, hindi, holandés, indonesio, inglés (Estados Unidos), inglés (India), inglés (Reino Unido), italiano, japonés, polaco, portugués (Brasil) y ruso.	Muy bueno. Voz muy natural	Alta definición	Google Play https://play.google.com/store/apps/details?id=com.google.android.tts&hl=es	4.0.3 y superiores	App 12,8 MB Voces 7,5 - 30 MB	Gratuita
SVOX Classic Text To Speech Engine	3.1.5	SVOX Mobile Voices	29 - Árabe (hombre), inglés australiano (mujer), portugués brasileño (mujer), francés canadiense (hombre/mujer), cantonés (mujer), checo (mujer), danés (mujer), holandés (hombre/mujer), finés (mujer), francés (hombre/mujer), alemán (hombre/mujer), griego (mujer), húngaro (mujer), italiano (hombre/mujer), japonés (mujer), coreano (mujer), mandarín (mujer), español mejicano (hombre/mujer), noruego (mujer), polaco (mujer), portugués (hombre/mujer), ruso (hombre/mujer), eslovaco (mujer), español (hombre/mujer), sueco (mujer), tailandés (mujer), turco (hombre/mujer), inglés Reino Unido (hombre/mujer), inglés Estados Unidos (hombre/mujer).	Malo. Voz robotizada	Estándar	Google Play https://play.google.com/store/apps/details?id=com.svox.classic	2.1 - 4.1	App 0,92 MB Voces ~18 MB	2,99 € / voz
IVONA Text-to-Speech HQ	1.6.42.524	IVONA Text-to-Speech	13 - Inglés (Estados Unidos), inglés (Reino Unido), inglés (Australia), francés, alemán, islandés, italiano, polaco, rumano, español, español (Estados Unidos), galés, galés inglés.	Muy Bueno. Voz bastante natural	Alta definición	Google Play https://play.google.com/store/apps/details?id=com.ivona.tts	1.6 - 5.0	App 501 KB Voces ~121 MB	Gratuita (beta pública)
ETI-Eloquence TTS	1.2.0	Eloquence	10 - Alemán, español (España), español (México), finlandés (Finlandia), francés (Canadá), francés (Francia), inglés (USA), inglés (UK), italiano y portugués (Brasil).	No probada, sin versión demo	No probada, sin versión demo	Google Play https://play.google.com/store/apps/details?id=es.codefactory.eloquencetts	4.0 y superiores	App 7,37 MB	19,99 € App y voces
Acapela TTS Voices	4.0.0.6	Acapela Group S.A.	31 - Árabe, catalán, chino mandarino, checo, danés, holandés (B), holandés (NL), inglés (Australia), inglés (India), inglés (Reino Unido), inglés (Estados Unidos), inglés (Escocia), finés, francés, francés (B), francés (Canadá), alemán, griego, italiano, japonés, coreano, noruego, polaco, portugués, portugués (Brasil), ruso, español, español (Norte América), sueco, sueco (Finlandia), turco.	Bueno	Estándar	Google Play https://play.google.com/store/apps/details?id=com.acapelagroup.android.tts	2.2 y superiores	App 3,16 MB Voces 20 -30 MB	App gratuita 3,99 € / voz

Tabla 10. Características de los principales motores de síntesis de texto a voz en Android

A la vista de los datos proporcionados, únicamente existen dos motores totalmente gratuitos, *IVONA Text-to-Speech* y *Síntesis de voz de Google*. El resto requieren de un pago por cada voz descargada (asociada a idioma y género) o bien un único pago para acceder a la aplicación y a las voces disponibles. En cuanto a la compatibilidad de idiomas, todos disponen de un amplio número de voces soportadas, llegando a 31 en el caso de *Acapela TTS Voices*.

En cuanto al rango de versiones de Android compatibles, todos los motores soportan las últimas versiones excepto *SVOX Classic* que establece Android 4.1 como la última disponible. En el lado opuesto, los dos únicos motores que no soportan versiones antiguas (1.x o 2.x) son *Síntesis de voz de Google* y *ETI-Eloquence TTS*, fijando en 4.0 la versión mínima necesaria.

Las dos características más importantes del análisis realizado en la Tabla 10, medibles por el **realismo** y **calidad de la voz**, establecen a la *Síntesis de voz de Google* y *a IVONA Text-to-Speech* como los motores que consiguen una voz más realista y aproximada al lenguaje natural, gracias a la pronunciación, entonación y calidad de las voces en alta definición. Estos dos motores se desmarcan del resto gracias a su naturalidad, pero atendiendo al tamaño de descarga de las voces de cada uno, 7,5MB y 30MB para el desarrollado por Google y cerca de 120MB para IVONA, se decide utilizar la ***Síntesis de voz de Google*** en colaboración con la aplicación desarrollada en este Proyecto Fin de Carrera.

2.5.2 Librerías TTS Android

Android facilita mediante su SDK, a partir del nivel 4 de API, el paquete ***android.speech.tts*** [43] para poner a disposición de los desarrolladores un motor de síntesis que permia hablar a las aplicaciones y establecer la forma en que lo hacen.

El motor TTS necesita conocer el lenguaje en el que debe hablar, ya que una misma palabra puede recibir distintas pronunciaciones dependiendo de la lengua elegida. Una vez instanciado el sintetizador en la aplicación, es indispensable establecer el lenguaje y opcionalmente la variante de esa lengua, antes de introducir la cadena de texto a pronunciar por el motor de síntesis.

A modo de resumen, la Tabla 11 muestra las descripciones de cada uno de los componentes del paquete *android.speech.tts.*

Tipo	Componente	Descripción
Interfaz	SynthesisCallback	Recoge la síntesis de voz procesada por el motor a partir de la cadena de texto proporcionada.
	TextToSpeech.OnInitListener	Define la interfaz a implementar para atender el evento lanzado al finalizar la inicialización del motor TTS.
	TextToSpeech. OnUtteranceCompletedListener	Interfaz deprecada en el nivel 18 de API, utilizar clase Utterance ProgressListener en su lugar.
Clase	SynthesisRequest	Gestiona los datos necesarios por el motor para realizar la síntesis: ▪ Texto a sintetizar. ▪ Lenguaje, país y variante del texto. ▪ Nombre de la voz a utilizar asociada al motor. No se utiliza si se ha indicado el punto anterior. ▪ Velocidad y tono de la voz.
	TextToSpeech	Sintetiza la voz para reproducirla de inmediato o crea un fichero de sonido, a través de la cadena de texto proporcionada. Es necesario esperar a la inicialización del motor antes de comenzar con la síntesis, implementando el interfaz *TextToSpeech.OnInitListener* permite descubrir el momento en el que se encuentra activo.
	TextToSpeech.Engine	Gestiona las constantes y parámetros utilizados para controlar el motor TTS. Son utilizados por *intent* que comprueban y/o instalan los datos del motor TTS en caso de ser necesario, por claves e identificadores que acompañan a los comandos de voz y una serie de propiedades soportadas por el motor de síntesis.
	TextToSpeech.EngineInfo	Información sobre el motor TTS instalado.
	TextToSpeechService	Clase base abstracta para implementar el motor TTS. Los métodos a implementar son: ▪ *onIsLanguageAvailable(String,String,String)* ▪ *onLoadLanguaje(String,String,String)* ▪ *onGetLanguage()* ▪ *onSynthesizeText(SynthesisRequest, SynthesisCallback)* – Es el método encargado de proveer al motor la cadena de texto en el objeto *SynthesisRequest* y devolver la síntesis resultante a través del objeto *SynthesisCallback*. ▪ *onStop()* – Indica al motor que debe finalizar cualquier síntesis que esté siendo reproducida.
	UtteranceProgressListener	Maneja los eventos relacionados con el progreso del habla dentro de la cola con las cadenas de texto pendientes de sintetizar.
	Voice	Describe las características de una voz TTS en términos de calidad y latencia.

Tabla 11. Componentes del paquete android.tts.speech

2.6 Enfermedad de Alzheimer: Diagnosis y terapia

2.6.1 Introducción

En primer lugar se define demencia como el deterioro de las capacidades mentales o intelectuales, como la memoria, la orientación, el lenguaje o la atención de un individuo, dificultando la realización de sus tareas cotidianas.

La enfermedad de Alzheimer, identificada en 1906 por el neurólogo alemán Alois Alzheimer, es el tipo más común de demencia que puede diagnosticarse. Se clasifica dentro de las neurodegenerativas primarias, que producen una degeneración de las células del sistema nervioso central [45]. El progreso de la enfermedad es lento y se caracteriza en su primera fase por la dificultad para retener nueva información.

Existen cuatro síntomas clínicos que definen el Alzheimer, "las cuatro A" [46]:

- **Amnesia:** es la pérdida en cualquier grado de la memoria, en un comienzo de la inmediata y finalmente de la memoria a largo plazo.

- **Apraxia:** es la incapacidad de realizar actos voluntarios aprendidos mediante función motora, como por ejemplo la imposibilidad de vestirse por sí mismo.

- **Agnosia:** disminución de la capacidad para reconocer las cosas y las personas.

- **Afasia:** problemática para nombrar correctamente los objetos cotidianos.

La Organización Mundial de la Salud (OMS) estima que existen 47,5 millones de personas en el planeta con algún tipo de demencia, número que crece cada año en 7,7 millones. El Alzheimer supone entre un 60% y 70% de esos casos. Los estudios epidemiológicos establecen que las personas a partir de 60 años tienen entre un 5% y 8% de probabilidades de sufrir una demencia [47].

En España el aumento de la esperanza de vida experimentado en las últimas décadas, ha propiciado un creciente número de pacientes con algún tipo de demencia. El colectivo con una edad superior a 80 años es el que ha experimentado un mayor crecimiento, siendo a su vez el más propicio a desarrollar la enfermedad. El grupo de población con más de 65 años (16,5% del total en 2008) arroja una prevalencia de enfermedad de Alzheimer comprendida entre el 5 y el 14,9%, aumentando en los mayores de 70 años a un rango entre 9 y 17,2%. Con los datos anteriores y realizando una media, puede estimarse una prevalencia de la enfermedad en torno al 7%, que supondría la existencia en España de 500.000-750.000 enfermos de Alzheimer. No existen cifras definitivas debido al desconocimiento del número de casos en estadios tempranos o no detectados [48].

2.6.2 Diagnóstico de la enfermedad

Existen múltiples test y escalas diseñados con el propósito de realizar un diagnóstico de la demencia, lo que demuestra la dificultad que supone estimar el grado concreto en un paciente. En el caso del Alzheimer, el conjunto de criterios más extendidos son los del **NINCDS/ADRDA**, que clasifican la enfermedad en tres niveles de certeza: posible, probable y definitiva [46].

En la Tabla 12 se describen los criterios de cada uno de los niveles. Para el diagnóstico de Alzheimer definitivo es necesario cumplir cada uno de los criterios de enfermedad probable y existir además una evidencia histológica mediante biopsia.

ENFERMEDAD DE ALZHEIMER PROBABLE
1. Demencia sugerida por examen clínico, documentada por un test tipo MMSE y confirmado por test neuropsicológicos.
2. Más de dos áreas cognitivas afectadas.
3. Déficit progresivo de memoria y otros ámbitos cognitivos.
4. Sin alteración de la conciencia.
5. Edad comprendida entre 45 y 90 años, siendo más habitual a partir de los 65 años.
6. Inexistencia de otros trastornos que pudieran explicar el declive cognitivo.
DIAGNÓSTICO APOYADO POR
1. Existen funciones cognitivas concretas afectadas: agnosia, apraxia, afasia.
2. Imposibilidad de realizar tareas habituales así como alteraciones de la conducta.
3. Historia familiar, en especial si se confirma a nivel neurológico.
4. Resultados de punción lumbar normal, atrofia visualizada a través de tomografía (TC).
ENFERMEDAD DE ALZHEIMER POSIBLE
1. Demencia sin existencia de otras enfermedades neurológicas o psiquiátricas que puedan causarla.
2. Existe otra enfermedad neurológica factible de causar demencia, pero no se considera causante de ella.
ENFERMEDAD DE ALZHEIMER DEFINITIVA
1. Cumple los requisitos de enfermedad probable.
2. Evidencia histológica a través de biopsia o necropsia.

Tabla 12. Niveles de certeza de la enfermedad de Alzheimer a través de los criterios diagnósticos NINCDS/ADRDA [46]

Otra escala muy difundida para cuantificar la degeneración neurológica es la llamada Global Deterioration Scale o **GDS** [49], que divide en siete fases el deterioro cognitivo. Parte de la fase 1, en la que no se aprecia ningún declive, pasando por estadios intermedios hasta llegar al 7 que corresponde a un declive severo. En cada una de las fases describe una serie de síntomas y señales característicos. El GDS suele acompañarse de una escala auxiliar que describe el deterioro funcional, denominada Functional Assesment Stage o **FAST** [50] y de la **BCRS** (Brief Cognitive Rating Scale) [51] con el

propósito de evaluar el deterioro de ámbitos cognitivos concretos. La enfermedad de Alzheimer suele asociarse a partir de un estadio GDS 4, es decir, con un déficit cognitivo moderado.

A modo de ejemplo, se muestra en la Tabla 13 una correlación de la GDS con la BCRS interpretada por Lluís Tarraga [52] con el propósito de facilitar la interpretación de los síntomas, añadiendo ámbitos cognitivos como la memoria (de fijación y evocada), orientación, concentración, lenguaje, cálculo y praxis constructiva.

Estadio	Concentración	Memoria fijación	Memoria evocación	Orientación	Lenguaje	Praxis	Cálculo
1 Ausencia de déficit cognitivo	Sin déficit objetivo ni subjetivo.	Sin déficit objetivo ni subjetivo.	Sin déficit objetivo ni subjetivo.	Sin déficit objetivo ni subjetivo.	Sin déficit objetivo.	Sin déficit objetivo ni subjetivo.	No se aprecia déficit objetivo ni subjetivo.
2 Déficit muy leve	Déficit subjetivo. Fácil distracción.	Déficit subjetivo.	Déficit subjetivo, es capaz de recordar el nombre de varios maestros de la escuela.	Déficit subjetivo, es capaz de indicar la situación y horas actuales.	Déficit subjetivo para nombrar personas y objetos.	Capacidad para dibujar un cubo.	Resta correctamente 43 menos 17.
3 Déficit leve	Dificultad para realizar series de 7 a partir de 100.	Déficit para recordar sucesos concretos pero recuerdos importantes intactos.	Indicios de déficit, recuerda un único maestro y/o amigo de su infancia.	Indicación de hora errónea en dos horas o más, de día actual en uno o más días, y en más de dos días con respecto al día del mes.	Déficit evidente en la elección de las palabras y ligero tartamudeo.	No puede dibujar un cubo correctamente.	Resta correctamente 39 menos 14.
4 Déficit moderado	Dificultad para realizar series de 7.	Olvida hechos importantes acontecidos en un corto periodo de tiempo (una semana).	Déficit evidente. Distorsión de la localización cronológica. Recuerda nombre de su escuela pero ningún profesor.	Indicación errónea de día actual en diez o más días, y de uno o más meses con respecto al actual.	Dificultad para pronunciar y posible tendencia a divagar.	Capacidad para dibujar un rectángulo.	Resta correctamente 15 menos 6.
5 Déficit moderadamente grave	Déficit evidente. Por ejemplo al enumerar los meses hacia atrás.	Afectación leve de orientación espacio-temporal.	Nulos recuerdos de sucesos marcados del pasado, como escuela, servicio militar.	Desorientación espacial actual. Incertidumbre en cuanto al día, mes y estación actuales.	Lenguaje espontáneo muy limitado. Capacidad para recitar un refrán.	Capacidad para dibujar dos circunferencias concéntricas.	Resta correctamente 9 menos 4.
6 Déficit grave	Olvida la orden. Por ejemplo al pedir contar del 1 al 10, realiza la orden al revés.	Existe algún recuerdo reciente. Desorientación espacio-temporal aguda.	Vestigios de memoria. Recuerda algunos datos como su nacionalidad, profesión, incluso nombre de los padres.	Desorientación temporal, reconoce a su pareja pero no su nombre. Recuerdo su propio nombre.	Frases de una o pocas palabras. Imposibilidad para recitar un refrán.	Capacidad para dibujar circunferencia, una línea y garabatos.	Suma correctamente: 8 más 7 y/o 3 más 1.
7 Déficit muy grave	Dificultad evidente para contar del 1 al 10.	Sin recuerdo de hechos acontecidos recientemente.	Ninguna evidencia de memoria.	Inseguridad en su nombre y no reconoce a su pareja.	Capacidad verbal limitada a una o dos palabras. Posibles gruñidos o gritos.	Incapacidad para dibujar nada. Puede sujetar un bolígrafo.	Suma en ocasiones 1 más 1 o no es capaz de realizar la suma.

Tabla 13. Correlación de la Global Deterioration Scale con el Functional Assessment Stage propuesta por Lluís Tarraga [52]

2.6.3 Técnicas terapéuticas

Las técnicas terapéuticas en pacientes de Alzheimer u otra demencia, se refieren a procesos de mantenimiento y estimulación de las funciones cerebrales, con el fin de crear nuevas conexiones y compensar las pérdidas. Un comienzo precoz de las técnicas y ejercicios propuestos, así como una correcta cualificación del profesional que los imparta, colaborará a proteger en mayor medida las capacidades neuronales y funcionales [46].

En la terapia se conjugan tanto tratamientos farmacológicos como no farmacológicos, para perseguir los siguientes objetivos:

- Aletargar el proceso de deterioro de las **funciones cognitivas** ya afectadas e intentar preservar las que se encuentran intactas.
- Diagnosticar prematuramente los posibles **trastornos emocionales** para poder establecer un tratamiento si fuera necesario.
- El fin último es el de mejorar la **calidad de vida** del enfermo y en extensión el de los cuidadores/familiares.

Dejando de lado la terapia farmacológica, la estimulación de las funciones cognitivas mediante un programa adecuado a los distintos niveles de deterioro (GDS) supone la línea más empleada por los terapeutas. A continuación se exponen algunos de los principales programas [53]:

- **Orientación de la realidad.** Trabaja la orientación de la persona mediante la interacción de información relacionada con el espacio, el tiempo y la propia persona
- **Reminiscencia.** Trabaja los recuerdos antiguos y sus localizaciones**.**
- **Estimulación y actividad cognitiva.** Trabaja distintos ámbitos cognitivos como la memoria, el reconocimiento, la comunicación verbal pero desde el punto de vista proactivo, en el que el paciente no sólo se limita a recibir estímulos sino que se implica activamente.
- **Terapia cognitiva específica.** Se centra en ciertos ámbitos cognitivos mediante el uso de ejercicios experimentales.
- **Instrumentos digitales para la ayuda cognitiva.** Existen novedosos entrenamientos cognitivos mediante la interacción con ordenadores y dispositivos electrónicos.

Adaptación

Independientemente de la terapia elegida, es una labor esencial realizar una adaptación apropiada de los ejercicios, estableciendo una dificultad acorde al GDS del paciente pero también buscando signos motivacionales en el contenido seleccionado. Un contenido inadecuado o una dificultad desacorde provocan una sensación de fracaso, cuando se pretende todo lo contrario, remarcar los éxitos de la persona. El Alzheimer posee un marcado rasgo heterogéneo, por lo que un tratamiento terapéutico homogéneo no sería una solución óptima.

2.6.3.1 Terapia de orientación

Su objetivo principal es trabajar los datos de ubicación espacial y temporal, así como la información personal del sujeto. A continuación se ejemplifican ejercicios tipo para este ámbito cognitivo:

- **Orientación temporal**
  - Realizar las siguientes preguntas:
    - ¿Qué hora es?
    - ¿Qué día de la semana es hoy?
    - ¿Qué estación del año es?
    - ¿Es de día o de noche?
- **Orientación espacial**
  - Realizar las siguientes preguntas:
    - ¿En qué ciudad estamos?
    - ¿En qué provincia estamos?
    - ¿En qué país estamos?
- **Orientación espacial**
  - Realizar las siguientes preguntas:
    - ¿Cuál es su nombre?
    - ¿Y sus apellidos?
    - ¿Cuál es su edad?
    - ¿Cuántos hijos tiene?

2.6.3.2 Terapia del lenguaje

Persigue trabajar alteraciones del lenguaje oral (afasias), del lenguaje escrito (agrafias) y la lectura (alexias). A continuación se ejemplifican ejercicios tipo para este ámbito cognitivo:

- **Lenguaje automático**
  - Realizar las siguientes indicaciones:
    - ¿Qué hora es?
    - ¿Qué día de la semana es hoy?
    - ¿Qué estación del año es?
    - Dígame los días de la semana.
- **Lenguaje espontáneo**
  - Realizar las siguientes indicaciones al mostrar una imagen:
    - ¿Qué se observa en esta lámina?
- **Denominación**
  - Realizar las siguientes preguntas al mostrar una lámina:
    - ¿Qué parte del brazo es ésta?
    - ¿Qué es este objeto?
    - ¿Qué hay en esta imagen?
- **Repetición**
  - Realizar las siguientes indicaciones:
    - Repita la siguiente frase: Iremos de vacaciones toda la familia.
    - Repita las siguientes palabras: bicicleta, cuchara, manzana.

- **Lectura - escritura**
  - Realizar ejercicios de lectura de palabras, frases, sílabas y letras.
  - Realizar ejercicios de escritura mediante redacción, dictado, copia, etc.
  - Completar palabras a las que le faltan letras o frases en las que faltan palabras.

2.6.3.3 Terapia de praxias

Esta terapia se centra en el entrenamiento de los movimientos organizados realizados para alcanzar un objetivo. La carencia en el control voluntario de los gestos o movimientos intencionados se denomina apraxia.

A continuación se ejemplifican ejercicios tipo para este ámbito cognitivo:

- **Praxias ideatorias**
  - Se proporciona al paciente una serie de objetos reales y se le solicita que los nombre.

- **Motricidad**
  - Seguir los puntos numerados por orden creciente para completar el dibujo.
- **Praxis constructiva**
  - Dibujar una serie de objetos.
    - Un coche.
    - Una bicicleta.
    - Un libro.
  - Terminar un dibujo que está incompleto.

2.6.3.4 Terapia de gnosias

Se define agnosia como la dificultad para reconocer sensorialmente nuestro entorno, sin que exista una afectación física de los órganos relacionados. En función del sentido afectado recibe una u otra denominación:

- Vista: agnosia visual.
- Oído: agnosia auditiva.
- Tacto: agnosia táctil.
- Olfato: anosmia.

A continuación se ejemplifican ejercicios tipo para este ámbito cognitivo:

- **Gnosia visual**
  - **Reconocimiento de imágenes.**
    - Mostrar una o varias fichas y realizar las preguntas necesarias:
      - ¿Qué animal ves en la lámina?
      - ¿Cuál es el nombre de este medio de transporte?
      - ¿Qué objeto observas en la imagen?

- **Reconocimiento de colores.**
  - Realizar las siguientes preguntas:
    - ¿De qué color es el cielo?
    - ¿De qué color es una pera?
  - Pedir que señale un color concreto dentro de una lámina.

- **Reconocimiento de caras.**
  - Dada una lámina:
    - ¿Cuál es el nombre de este actor?
    - ¿Cómo se llama este cantante?
    - ¿El niño está triste o contento?
  - Dada un conjunto de láminas:
    - Señale a un político.

2.6.3.5 Terapia de memoria

Mediante esta terapia se pretenden estimular los recuerdos del paciente, tanto los inmediatos (memoria inmediata), los recientes (memoria reciente) y aquellos que forman parte de nuestra experiencia de vida (memoria episódica). Cualquier indicio de disminución de la memoria se denomina amnesia.

A continuación se ejemplifican ejercicios tipo para este ámbito cognitivo:

- **Memoria inmediata y reciente**
  - **Imágenes**
    - Se muestra una fotografía personal o familiar:
      - ¿Cuál es el nombre de las personas que aparecen en la fotografía?
    - Se guarda la fotografía personal o familiar:
      - ¿Cuál es el nombre de las personas que recuerda de la fotografía?
  - **Verbal**
    - Se muestra un texto con ciertos datos importantes y se le realiza alguna pregunta relacionada con el contenido:
      - ¿En qué día partían de viaje?
      - ¿Cuál era el nombre de la mascota?
    - Se solicita la lectura de un conjunto de refranes y a continuación se solicita que repita cualquiera de ellos.

- **Memoria episódica**
  - Se realizan una serie de preguntas relativas a sucesos relevantes en su vida:
    - ¿En qué fecha se casó?
    - ¿Dónde fue de viaje de novios?
    - ¿Cuál fue su primer trabajo?
  - Se le solicita que recite un refrán.

2.6.3.6 Terapia de cálculo

Esta terapia se centra en la capacidad para realizar **operaciones aritméticas** (sumas, restas, multiplicaciones y divisiones), básicas en las tareas cotidianas del paciente como pueda ser hacer la compra. A la alteración de la capacidad para operar aritméticamente se le denomina **acalculia**.

A continuación se ejemplifican ejercicios tipo para este ámbito cognitivo:

- **Secuencias de números**
  - Clasificación de números pares e impares.
  - Ordenación numérica.
  - Progresión numérica.

- **Operaciones aritméticas**
  - Suma
    - Sume 43 y 27.
  - Resta
    - Reste 12 menos 6.
  - Multiplicación
    - Multiplique 4 por 12.
  - División
    - Divida 12 entre 3.

- **Reconocimiento**
  - Pedir la trascripción numérica de un conjunto de cifras:
    - Escriba noventa y tres.
    - Escriba ciento cuarenta y siete.
  - Pedir la trascripción alfabética de un conjunto de cifras:
    - Escriba 68.
    - Escriba 334.

2.7 Aplicaciones Android sobre Alzheimer

En las siguientes líneas se realiza un estudio de algunas aplicaciones destacables cuya temática principal es la enfermedad de Alzheimer y que se encuentran publicadas actualmente en Google Play.

Andzheimer [54]

El autor de esta aplicación informa sobre el propósito de ayuda en la estimulación de las capacidades cognitivas de los pacientes de Alzheimer, utilizando una serie de ejercicios gráficos. Se ha desarrollado siguiendo las guía de accesibilidad WCAG / AARP (Audience-Centered Heuristics: Older Adults) y NIH (National Institute on Aging). Además, el desarrollador insta a que la persona que guía al paciente en la realización de los ejercicios sea siempre la misma para poder comprobar la evolución y poder decidir si subir o bajar la dificultad de los mismos.

Al ingresar en la aplicación muestra una pantalla de selección de idioma, encontrándose disponibles inglés y castellano. Una vez seleccionado el idioma, una voz confirma nuestra elección mediante el uso de TTS.

Figura 28. Pantalla principal de Andzheimer

La pantalla principal (Figura 28) presenta un menú de ejercicios categorizado por ámbito cognitivo: memoria, atención, gnosias, praxias, lenguaje y ejecutivas. Al pulsar sobre el icono de una categoría aparecerá una nueva pantalla con la presentación visual del ejercicio precedido de una ficha técnica en la que se indica la función cognitiva a tratar así como unas breves instrucciones, las cuales podrán ir dirigidas al cuidador o al paciente. Cada acción realizada es informada verbalmente mediante TTS. La Figura 29 muestra un ejercicio sobre el reconocimiento de formas (Gnosias).

Figura 29. Ejercicio para el reconocimiento de formas en aplicación Andzheimer

Se detectan ciertas carencias o puntos a mejorar como la existencia de ejercicios únicos para cada ámbito cognitivo o la falta de contraste de datos correctos en ejercicios como el de memoria (Figura 30). Si se introducen datos erróneos en cualquiera o todos los campos a rellenar, la validación de la respuesta es correcta.

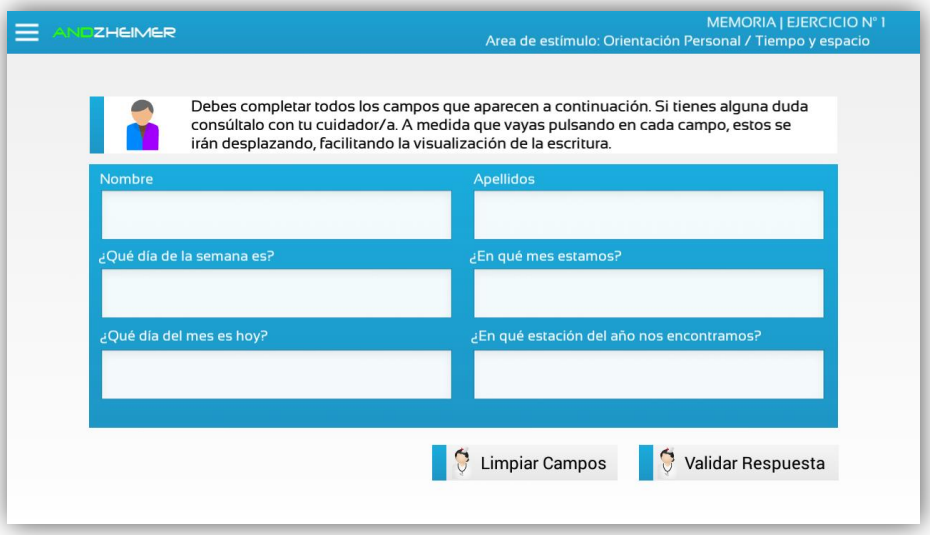


Figura 30. Ejercicio presentado por Andzheimer para fomentar la memoria

Tweri [55]

Aplicación de carácter gratuito desarrollada por Solusoft en colaboración con la Asociación de Familiares de enfermos de Alzheimer (AFAL) de Getafe - Madrid. Su finalidad consiste en la geolocalización continua (basada en GPS) de la persona afectada por la enfermedad, de tal modo que en caso de pérdida o desorientación pueda ser fácilmente localizada por el cuidador. El enfermo obtiene por tanto una mayor independencia para poder realizar paseos sin depender directamente de una segunda persona.

Una vez instalada, se establecen un área de seguridad y tiempo máximos a partir de los cuáles la aplicación alertará mediante el envío de un SMS al cuidador con la localización del paciente. También existe la posibilidad de realizar una llamada de emergencia al cuidador incluso estando dentro de los límites previamente fijados.

Debido al doble rol de uso de la aplicación, afectado y cuidador, la aplicación solicita el registro en función del tipo de usuario (Figura 31).

Figura 31. Pantalla de selección de rol de usuario (cuidador o afectado) en Tweri

Por la naturaleza de la aplicación, es primordial que el afectado conecte el GPS de su terminal y acceda a Tweri antes de salir de paseo, pero gracias a la implementación realizada en la parte servidor, si el móvil pierde la señal GPS, la conexión de datos o incluso se queda sin batería, el cuidador sería alertado al superar el intervalo de tiempo preestablecido.

Alzheimer Info y ejercicios [56]

La aplicación basa su funcionamiento en una serie de ejercicios basados en el MMSE de Folstein, con la finalidad de reforzar los ámbitos cognitivos trabajados y realizar una estimación del grado de la enfermedad (Figura 32).

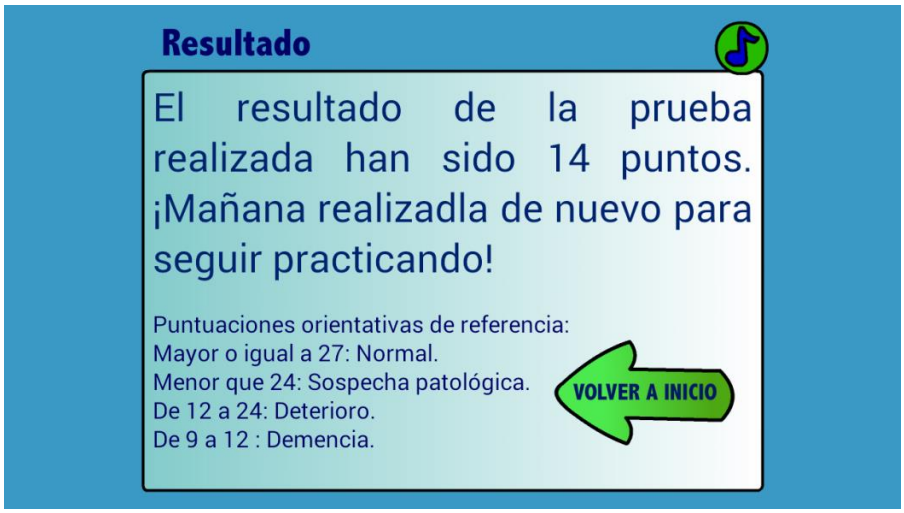


Figura 32. Pantalla de resultado del test MMSE en función de las respuestas del paciente

Según su planteamiento, se requiere la ayuda de un cuidador que explique y guie en la realización de los diversos ejercicios, quién además será el encargado de validar la respuesta del paciente en cada ficha, como puede observarse en la Figura 33.

Figura 33. Interfaz de ficha de ejercicio, el cuidador valida la respuesta

Mid-stage Alzheimer's Cards [57]

Esta aplicación es una herramienta de comunicación que facilita la interacción entre el enfermo de Alzheimer y su cuidador/familia. Mediante un conjunto de fichas categorizadas (Figura 34) el paciente puede expresar sus sentimientos, necesidades o requerimientos, seleccionando la que más se ajuste al mensaje que pretende transmitir. En la Figura 35 se observa a modo de ejemplo las fichas disponibles para seleccionar dentro de la categoría “tiempo”.

Figura 34. Pantalla de selección de categoría para las tarjetas de Mid-stage Alzheimer's Cards

Es especialmente concebida para los grados de enfermedad en los que la capacidad semántica se ha visto afectada o incluso impedida. No se encuentra disponible en castellano, se ha desarrollado íntegramente en inglés.

Figura 35. Ejemplo de fichas disponibles para el paciente en la categoría “tiempo”

[re]membr [58]

Aplicación desarrollada por la empresa tecnológica Supertruper con el propósito de mantener la independencia y calidad de vida del enfermo de Alzheimer en sus primeros estadios, gracias a la identificación de productos o lugares a través de un código inequívoco. Permite a un familiar o cuidador, asociar un mensaje de audio y una descripción textual a un producto escaneando su código de barras (Figura 36), de esa manera cuando el usuario lo escanee (Figura 37), recibirá información sobre qué es y para qué se utiliza. Para asociar un mensaje a un lugar como por ejemplo una habitación, será necesario disponer de un nuevo código para imprimirlo y pegarlo en la estancia deseada.

Figura 36. Pantalla principal de [re]membr con acceso a llamada de emergencia o escáner de códigos de barras

Otra funcionalidad destacada es la posibilidad de realizar una llamada de emergencia a un teléfono prestablecido para este caso, complementado además con un mensaje de texto y audio tranquilizador mientras se realiza la conexión telefónica, puede apreciarse el acceso a esta funcionalidad en la Figura 36.

Aunque gratuita, la aplicación permite asociar un límite de 50 códigos, siendo necesario pagar para conseguir códigos adicionales.

Figura 37. Proceso de escaneo del código de barras de un nuevo producto

2.7.1 Conclusiones aplicaciones Alzheimer

Una vez analizadas en el Apartado 2.7 las aplicaciones destinadas a la enfermedad de Alzheimer, se detectan de forma general las siguientes carencias o puntos mejorables:

- Falta de **métodos alternativos** de interacción con la aplicación, como por ejemplo por medio del lenguaje natural.
- Funciones de **pago**.
- Ausencia de implementación de **sistemas de diagnóstico** de la enfermedad en las aplicaciones.

Capítulo 3

Tecnologías y herramientas

Este capítulo realiza una descripción de las principales herramientas y tecnologías utilizadas dentro del sistema HealthCoach, exponiendo sus características y la funcionalidad aportada en el desarrollo del proyecto.

3.1 Tecnologías

3.1.1 XML

XML (*e**X**tensible **M**arkup **L**anguage*) es un meta-lenguaje desarrollado por el W3C (*World Wide Web Consortium*) [59] que juega un papel importante en el intercambio de datos dentro de la web o entre aplicaciones, definiendo la gramática para diseñar lenguajes de marcado como por ejemplo HTML. Sin embargo, el cometido de XML es describir datos, no el de mostrarlos, como sí es el propósito de HTML.

Los documentos XML son almacenados con extensión .xml y contienen un conjunto de datos con sus respectivas etiquetas de marcado, como puede observarse en el ejemplo de la Figura 38.

Para el desarrollo de las dos aplicaciones Android realizadas se utilizan archivos XML denominados ***layouts*** (Figura 38), que definen la interfaz gráfica con la que interactúa el usuario. Android permite también utilizar código Java para definir la interfaz, pero el uso de *layouts* basados en XML proporciona independencia entre funcionalidad y vista.

```
<?xml version="1.0" encoding="UTF-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/RelativeLayout1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
     android:orientation="vertical"
     android:layout_gravity="center_vertical"
    android:background="@drawable/blog_rounded_edges" >

    <ImageView
        android:id="@+id/image"
        android:layout_width="200dp"
        android:layout_height="100dp"
        android:layout_centerHorizontal="true"
        android:src="@drawable/user" />

    <ImageView
```

```
        android:id="@+id/imageLang"
        android:layout_width="100dp"
        android:layout_height="50dp"
        android:layout_marginTop="20dp"
        android:layout_alignParentRight="true"/>

    <TextView
        android:id="@+id/text1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/image"
        android:layout_centerHorizontal="true"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textSize="40dp"
        android:textColor="#000000" />

    <TextView
        android:id="@+id/text2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/text1"
        android:layout_centerHorizontal="true"
        android:textColor="#000000"
        android:textAppearance="?android:attr/textAppearanceLarge" />

</RelativeLayout>
```

Figura 38. Layout de Android como ejemplo de estructura de fichero XML

3.1.2 Java

El lenguaje de programación Java [60] es desarrollado por la compañía Sun Microsystems (posteriormente adquirida por Oracle) en 1991 pero publicado en 1995, en concreto por un equipo de programadores liderados por James Gosling. Hoy en día, es el lenguaje de programación más extendido a nivel mundial, principalmente en el uso cliente-servidor de aplicaciones web, con más de 9 millones de desarrolladores.

Gosling definió cinco objetivos fundamentales a la hora de diseñar el lenguaje Java: orientado a objetos, multiplataforma, soporte para trabajo en red, ejecución segura en sistemas remotos y la facilidad de uso basándose en las fortalezas de C++ pero sin sus utilidades a bajo nivel.

Las aplicaciones Java se encuentran compiladas en *bytecode* o fichero binario, son las llamadas clases con extensión .class, que puede ser interpretado por cualquier máquina virtual de Java (JVM) con independencia de la arquitectura de la máquina en la que se encuentra.

El sistema operativo Android mediante su SDK facilita la creación de aplicaciones desarrolladas en lenguaje Java, aunque utilizando su propia máquina virtual denominada Dalvik (explicada en el **Capítulo 4**).

3.1.3 HTTP

El Hypertext Transfer Protocol o protocolo de transferencia de hipertexto [61] es propuesto por Tim Berners-Lee para establecer un sistema global de intercambio de información en el *World Wide Web*, ya sean ficheros HTML, imágenes o cualquier otro recurso. Son precisamente esos recursos los que pueden ser identificados por una URL, *Uniform Resourse Locator* o Localizador de Recursos Uniforme en español.

HTTP define por tanto, las reglas para el intercambio de información entre un cliente y un servidor dentro del marco de la Web.

Una de las principales características del protocolo HTTP es su falta de estado, es decir, no guarda información de sesiones anteriores. Para suplir esta carencia los desarrolladores recurren con frecuencia a las cookies, se trata de información útil para el servidor que se almacena en el cliente por un tiempo indeterminado.

Con el primer estándar, HTTP/1.0, se definen tres comandos para realizar gestionar el envío y recepción de la información:

- **GET** – Utilizado para solicitar información al servidor, como sucede al pulsar en un enlace o introducir una URL en un navegador web. Es el comando HTTP más utilizado.
- **POST** – Permite enviar información al servidor, como los datos recogidos en un formulario.
- **HEAD** – Sirve para obtener los metadatos (tamaño, fecha de creación, etc.) de un objeto. Muy utilizado en la gestión de cachés de páginas de los navegadores y por los sistemas proxy.

En posteriores versiones se amplía hasta ocho el número de comandos soportados

- **PUT** – Permite subir un recurso completo al servidor.
- **DELETE** – Borra un recurso alojado en el servidor.
- **CONNECT** – Comprueba la conexión con el servidor.
- **TRACE** – Indica al servidor que devuelva la petición que ha recibido, se utiliza con propósitos de debug.
- **PATCH** – Para editar una parte concreta del recurso.

Un navegador Web es por tanto un cliente HTTP que envía peticiones de recursos (comandos GET) a un servidor HTTP (servidor Web), quién devuelve los recursos al cliente. Por defecto, el puerto en el que escuchan los servidores HTTP es el 80, aunque pueden configurarse para utilizar cualquier otro.

HTTP define códigos de estado para las comunicaciones entre servidor y cliente, compuestos por tres dígitos:

- **1xx** Mensajes informativos.
- **2xx** Éxito en la operación.
- **3xx** Redirección.
- **4xx** Error en el lado del cliente.
- **5xx** Error en el lado del servidor.

3.1.4 REST

REST proviene de *REpresentational State Transfer* [62] y se trata de una arquitectura para diseñar aplicaciones que se comuniquen en la Web utilizando un sencillo mecanismo basado en el protocolo HTTP. La principal utilidad de REST es la de implementar servicios Web, cuando un servicio o API utiliza la arquitectura REST se denominan ***RESTful*** [63].

Debido a que REST se basa en el estándar HTTP, es indispensable conocerlo en profundidad a la hora de desarrollar un servicio RESTFul. Aspectos claves son los métodos, códigos de error y tipos de contenido HTTP, además de una correcta utilización de las URIs, las cuáles deben seguir unas normas básicas a la hora de identificar un recurso:

- No deben contener nombres que impliquen una acción, es decir, debe evitarse el uso de verbos.

```
GET /usuario/3
```

- Al tratarse de un identificador debe de ser único.
- Seguir una jerarquía lógica.
- Independientes del formato del recurso que representen, como pueda ser XML, JSON, PDF, etc.
- No deben incluir filtros de información.

```
GET /usuario?nombre=David&apellido=López
```

3.1.5 JSON

El formato JSON (*JavaScript Object Notation*) [64] se utiliza para realizar intercambios de información a través de una sencilla sintaxis y compatible con todos los lenguajes de programación. Surge como alternativa a XML y ha sido muy aceptado por la comunidad de desarrolladores debido a su fácil interacción a través de JavaScript.

Existen dos tipos de estructuras dentro de la sintaxis propia de JSON:

- **Objetos JSON**. Se definen entre llaves y están formados por pares de clave-valor separados por una coma, usando dos puntos para asociar el valor a la clave. A continuación se muestra un ejemplo que contiene datos de usuario de HealhCoach:

```
{"NAME":"David","EMAIL":"admin@admin.es"}
```

- **Arrays JSON**. Son colecciones de objetos JSON separados por comas y delimitados de forma global por corchetes, como muestra el siguiente ejemplo extraído de un fragmento JSON de usuarios de HealhCoach:

```
[{"IDU":"1","NAME":"Marcos","EMAIL":"admin@admin.es","IMEI":"86509202351
5663","LEVEL":"7","GENDER":"F","TYPE":"P","LANGUAGE":"ESP"},
{"IDU":"2","NAME":"David","EMAIL":"admin@admin.es","IMEI":"8650920235156
62","LEVEL":"6","GENDER":"M","TYPE":"P","LANGUAGE":"ESP"}]
```

Es habitual que las aplicaciones formateen la información utilizando los dos objetos que se acaban de describir. Continuando con la aplicación HealthCoach, un JSON real generado para devolver los usuarios de la aplicación devolvería un objeto de tipo *users* cuyo valor sería un array JSON con objetos que representan los datos de cada usuario:

```
{"users":
[{"IDU":"1","NAME":"Marcos","EMAIL":"admin@admin.es","IMEI":"865092023515663",
"LEVEL":"7","GENDER":"F","TYPE":"P","LANGUAGE":"ESP"},
{"IDU":"2","NAME":"David","EMAIL":"admin@admin.es","IMEI":"865092023515662",
"LEVEL":"6","GENDER":"M","TYPE":"P","LANGUAGE":"ESP"}]}
```

3.1.6 PHP

PHP (*PHP Hypertext Pre-processor*) [65] es un lenguaje de programación de propósito general y de código abierto, muy utilizado en el desarrollo web para generar contenido HTML de forma dinámica. Su ejecución se realiza en el servidor, que mediante un módulo capaz de interpretar código PHP, genera el contenido HTML definitivo que será enviado de vuelta al cliente.

Su sintaxis se inspira en componentes de Java, C y Perl. Los interpretes de PHP sólo ejecutan el código comprendido entre los delimitadores *<?php* y *?>*, que distinguen el código PHP del resto, como puede ser HTML. Otra característica sintáctica es la definición de variables por medio del símbolo del dólar $ sin la necesidad de indicar el tipo. Como sucede en los lenguajes en los que se basa, cada sentencia debe terminar en punto y coma.

El siguiente código PHP realiza una conexión con base de datos MySQL y recupera información acerca de la entidad *USERS*.

```
<?php
$username = "X";
$password = "X";
$hostname = "192.168.1.X";

//connection to the database
$dbhandle = mysql_connect($hostname, $username, $password)
 or die("Unable to connect to MySQL");
echo "Connected to MySQL<br>";

//select a database to work with
$selected = mysql_select_db("proyectofincarrera",$dbhandle)
  or die("Could not select examples");

//execute the SQL query and return records
$result = mysql_query("SELECT IDU, NAME, AGE FROM USERS");

//fetch tha data from the database
while ($row = mysql_fetch_array($result)) {
   echo "IDU:".$row{'IDU'}." Nombre:".$row{'NAME'}." Edad:
```

```
".$row{'AGE'}."<br>";
  }
  //close the connection
  mysql_close($dbhandle);
  ?>
```

Adicionalmente, a partir de la versión 5 de PHP se habilita para los desarrolladores la interfaz PDO (*PHP Data Objects*) [66] para facilitar la conexión con distintos tipos de bases de datos, suministrando controladores específicos para cada una de ellas. Esta interfaz abstrae la comunicación con la base de datos utilizando siempre las mismas funciones con independencia de la base de datos existente.

3.1.7 MySQL

MySQL [67] es el sistema de administración de bases de datos (Database Management System o DBMS) de código abierto y gratuito (en su versión *community)*, más utilizado a nivel mundial en desarrollos orientados a internet. Su nombre proviene de *My Structured Query Language* y aunque originalmente desarrollado por MySQL AB, en 2008 pasa a ser propiedad de la multinacional Oracle.

MySQL está desarrollado en C/C++, proporcionando una gran estabilidad y alta velocidad de respuesta. Cuenta además con soporte para multiprocesadores.

Una de las principales ventajas de este gestor de bases de datos relacionales es su carácter multiplataforma (Windows, Unix, Mac, etc.) y la fácil interacción con lenguajes como Java, PHP o Perl.

Con la versión 5.0 se solventa uno de los inconvenientes de este DBMS, la falta de procedimientos almacenados, funciones, disparadores y vistas. Los procedimientos almacenados y funciones agrupan conjuntos de comandos SQL en el servidor que pueden ejecutarse con una sola invocación, mientras que los disparadores o *triggers*, son objetos asociados a una tabla de la base de datos que permiten realizar una acción al producirse un evento dentro de ella. Una vista es un objeto que se define en el servidor a partir de una consulta, para poder referirse a ella en sucesivas ocasiones de forma simplificada.

La base de datos del sistema HealthCoach que contiene los datos de los usuarios, tarjetas y log de respuestas de la terapia se ha generado gracias a una instancia MySQL, cuyo manual de instalación puede observarse en el **Anexo A**.

3.1.8 Apache HTTP

Desarrollado por la *Apache Software Foundation*, el servidor Apache HTTP [68], más conocido como Apache, es un servidor de código abierto y gratuito para servir páginas y servicios HTTP. Por lo tanto, es un programa diseñado para transferir datos de hipertexto cuando una aplicación cliente realiza una petición al servidor web, respondiendo con información estructurada en formato HTML a través de la red.

Entre sus ventajas se encuentran la facilidad de instalación y configuración, su distribución gratuita, el carácter multiplataforma siendo compatible con Linux, Windows y MacOS, soporte de la comunidad de programadores al tratarse del servidor de aplicaciones más extendido en el mundo, un gran rendimiento y seguridad implementada mediante los protocolos SSL y TLS.

Otra característica destacable es la modularidad de Apache, basada en el núcleo que aporta la funcionalidad principal, se complementa de una serie de módulos para ampliarla. En referencia a este Proyecto Fin de Carrera, el uso del módulo *mod_php* posibilita la ejecución de código escrito en lenguaje PHP.

La configuración de Apache se realiza mayoritariamente a través de los ficheros *apache2.conf* o *httpd.conf* en función del sistema operativo donde se aloje el servidor. Para que cualquier cambio sea efectivo requiere reiniciar el servidor.

3.1.9 Slim

Slim [69] es un pequeño framework PHP destinado a crear aplicaciones web y APIs REST de forma rápida gracias a su sintaxis simplificada, que soporta tanto métodos HTTP estándar como personalizados, redirecciones de rutas, gestión de errores, etc.

3.1.10 JavaMail-Android

JavaMail para Android [70] es una adaptación a esta plataforma de la API JavaMail originariamente desarrollada para el envío de correos electrónicos desde aplicaciones Java. Su utilidad reside en la capacidad de componer y enviar los correos electrónicos a través de código sin requerir intervención del usuario. El SDK de Android únicamente permite invocar a la aplicación de correo integrada en el sistema para realizar la redacción manual.

3.1.11 AndroidPlot

AndroidPlot [71] es una API gratuita y de código abierto, destinada a la creación de gráficas estáticas y dinámicas en aplicaciones Android. Ofrece soporte para gráficos de líneas, de barras, de tarta, etc.

3.2 Herramientas

3.2.1 JDK

Java Development Kit [72] es el entorno de desarrollo proporcionado por la compañía Oracle para construir aplicaciones en lenguaje Java. Incluye el *Java Runtime Environment* o *JRE*, el compilador y las APIs de Java.

La instalación del JDK es imprescindible en la realización de este proyecto al ser un requisito para utilizar el SDK de Android. En el siguiente apartado se explica el proceso de instalación.

Instalación

En entornos Windows, la instalación del JDK se realiza con estos pasos:

1. Descargar la versión instalable de 32 o 64 bits disponible dentro del portal de Oracle [73]. Una vez descargado se ejecuta el fichero ejecutable para proceder a su instalación.
2. Configurar la variable de entorno del sistema PATH para indicar la localización de los dos ficheros ejecutables Java, el compilador *javac.exe* y el intérprete *java.exe*. Desde Panel de control>Sistema y Seguridad>Sistema>Configuración avanzada del sistema. Desde esa ventana se selecciona en la parte inferior el botón "Variables de entorno". Se mostrará otra ventana donde es necesario buscar en la sección "Variables del sistema", la variable "Path". Se selecciona y pulsa en "Editar" para añadir dentro de "Valor de la variable" y a continuación del contenido que ya exista la ruta a la carpeta bin de nuestro JDK, precedida de un punto y coma. La Figura 39 muestra un ejemplo de ruta para la versión 1.7.

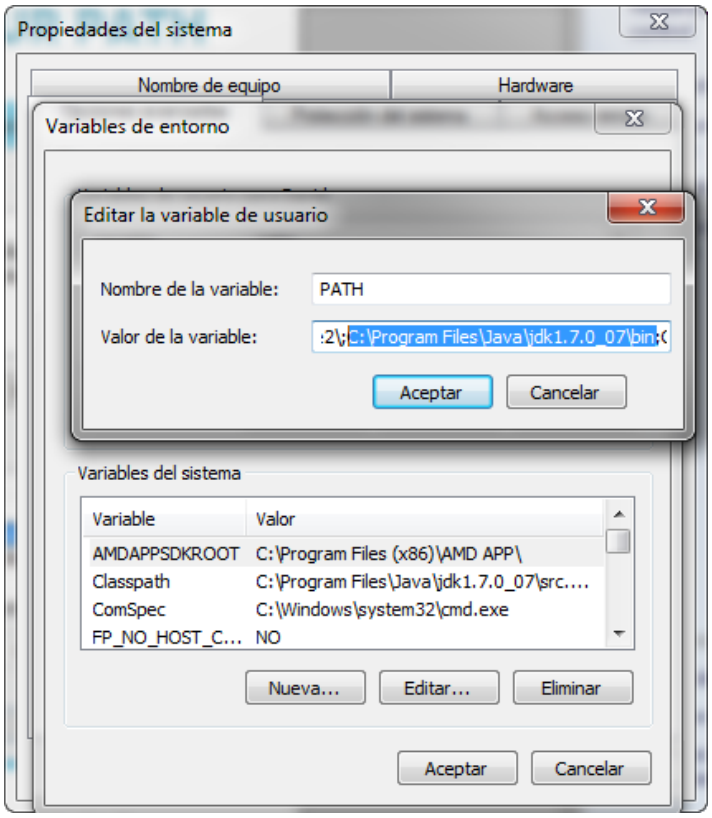


Figura 39. Configuración de la variable de entorno PATH

3.2.2 Eclipse

Como *IDE* o entorno de desarrollo integrado se ha elegido Eclipse de entre las opciones disponibles. Además de poseer una licencia de código abierto y ser multiplataforma, existen dos razones que fundamentan la elección, por un lado facilita el desarrollo de las dos aplicaciones Android gracias al soporte de Google a este entorno con la creación del plugin ADT (*Android Development Tool*) y por otro la experiencia previa con este entorno.

Gracias a la funcionalidad añadida con el complemento ADT, Eclipse es capaz de crear la estructura de un proyecto Android, compilarlo, realizar *debug* del código e incluso generar el apk firmado de la aplicación para su distribución en Google Play.

3.2.2.1 Instalación de Eclipse

El proceso de instalación de Eclipse en entornos Windows es muy sencillo, basta con descargarse la última versión del IDE desde la sección de descargas de su página oficial [74]. Para la realización de este proyecto se ha utilizado la versión 4.2.1 denominada Juno, una vez descargado el fichero .zip se procede a descomprimir el contenido en la carpeta donde queramos que permanezca la instalación. Dentro se encuentra el ejecutable Eclipse.exe responsable de arrancar el programa.

La primera vez que arranca el IDE solicita al usuario una dirección del directorio de trabajo o *workspace* donde se guardarán los proyectos generados, como muestra la Figura 40.

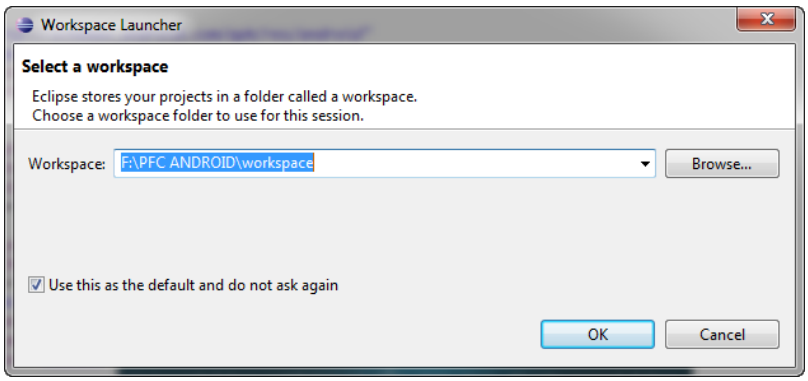


Figura 40. Selección de la ruta de workspace en Eclipse

3.2.3 SDK Android

Google pone a disposición de los desarrolladores el SDK (Software Development Kit) de la plataforma Android [75], con la finalidad de distribuir la API, documentación y herramientas de Android en sus distintas versiones. Cada versión del sistema operativo conlleva una nueva versión del SDK, en ocasiones es necesario recurrir a versiones previas por razones de compatibilidad. Sin esta herramienta no sería posible compilar las clases Android y crear las aplicaciones ejecutables de extensión .apk listas para instalar en terminales Android compatibles.

Como requisito previo a su uso, deberá instalarse la versión 7 del JDK en la máquina que alojará el SDK.

La descarga se realiza desde la página habilitada a tal efecto por Google [76], dentro de la sección *SDK Tools Only* aparecen las plataformas y las opciones de descarga disponibles. En el caso de Windows existe un ejecutable y un archivo comprimido en formato .zip.

Una vez descargado el SDK será necesario descomprimirlo en la carpeta que se desee, no es relevante la ubicación pero debe recordarse para establecerla en el plugin ADT de Eclipse. Al realizar esta asociación puede arrancarse el *SDK Manager* desde *Window > Android SDK Manager*, otra manera de acceder a esta interfaz desde Windows es ejecutando el fichero *SDK Manager.exe* de la carpeta que contiene la instalación.

SDK Manager separa en paquetes las herramientas, APIs, códigos de ejemplo e imágenes virtuales de dispositivos que están disponibles para su descarga. Además contiene un módulo de actualización que comprueba si existen revisiones de los componentes ya descargados, como puede observarse en la Figura 41.

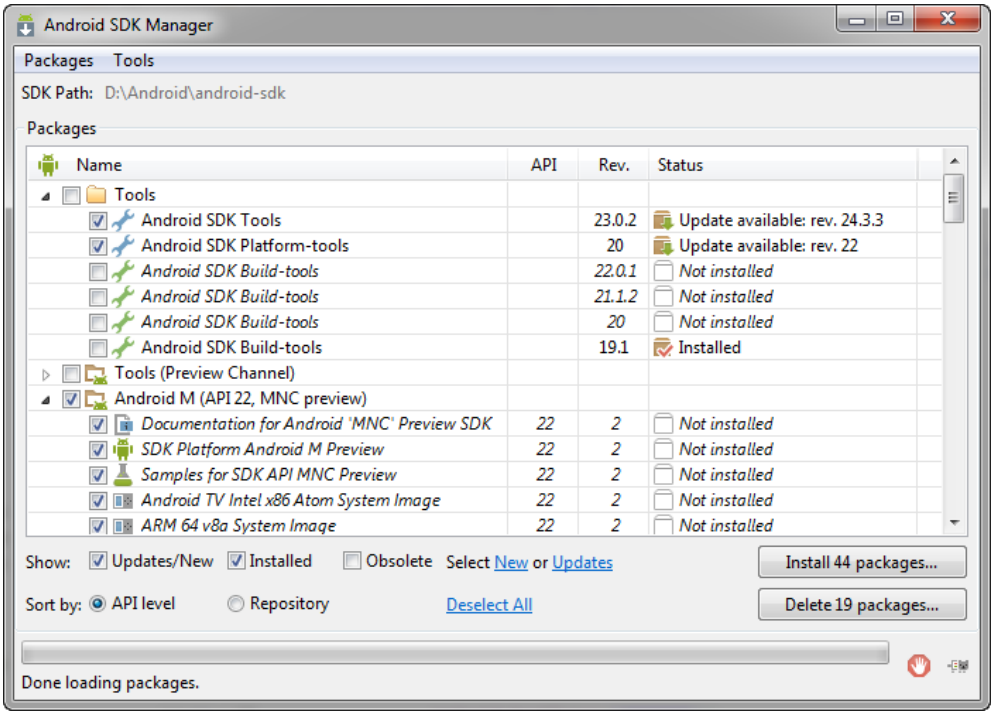


Figura 41. Interfaz del SDK Manager de Android

3.2.3.1 Plugin ADT

La herramienta ADT (*Android Development Tools*) [77] proporciona mediante su integración con Eclipse en forma de plugin, los componentes necesarios para realizar desarrollos en la plataforma Android de forma rápida y sencilla. Es el mecanismo utilizado para realizar la depuración, compilación y creación de ejecutables en formato apk para su instalación en dispositivos reales.

Instalación

Para realizar la instalación de ADT en Eclipse deben seguirse estos pasos:

1. Pestaña *Help> Install New Software*.
2. En la nueva ventana, pulsar en al botón *Add…* de la derecha.
3. En la ventana Add Site mostrada en la Figura 42, rellenar los campos *Name* y *Location* con los valores:
   - *Name*: ADT (o el nombre que se desee).
   - *Location*: https://dl-ssl.google.com/android/eclipse/
4. Pulsar *OK*.
5. Después de la búsqueda de actualizaciones, Eclipse mostrará que existen paquetes disponibles para Android Developer Tools, se seleccionan todos y se instalan
6. Reiniciar Eclipse para que el nuevo plugin esté disponible.

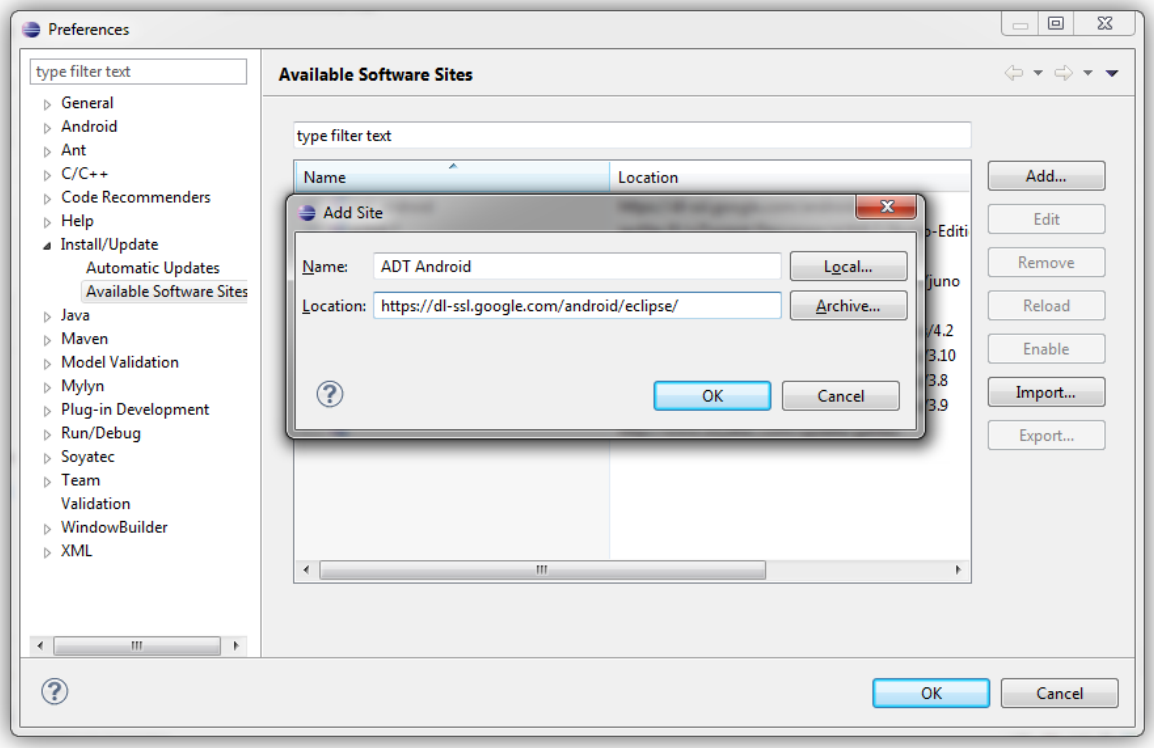


Figura 42. Repositorio de ADT en Eclipse

Referenciar SDK Android

Una vez instalado ADT es necesario establecer la localización del SDK de Android en nuestro sistema. Para ello, desde Eclipse se selecciona *Windows* > *Preferences*, desde la nueva ventana en el panel izquierdo *Android* y por último en *SDK Location* introducir la ruta del SDK (Figura 43) establecida en el **Apartado 3.2.3** (Figura 41).

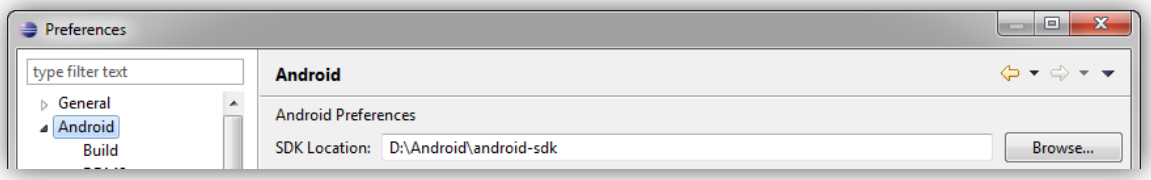


Figura 43. Ruta del SDK de Android dentro de Eclipse

3.2.3.2 Crear un proyecto Android

Gracias al plugin ADT pueden crearse proyectos Android de forma rápida desde Eclipse. Utilizando el asistente de creación de nueva aplicación Android (Figura 44) se construye la estructura mínima necesaria para empezar a desarrollar.

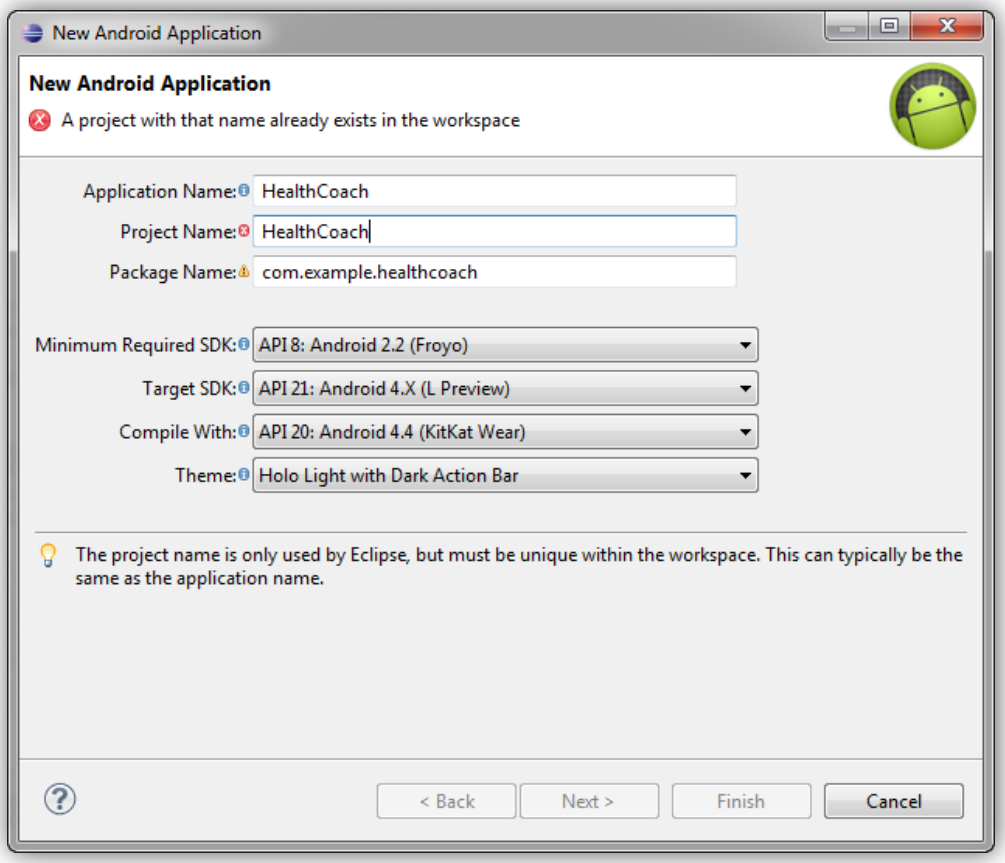


Figura 44. Pantalla de creación de nuevo proyecto Android en Eclipse

Los pasos necesarios para crear el proyecto son:

1. Seleccionar desde el menú *File> New > Android Application Project*.
2. Desde la ventana generada se introducen los campos mostrados en la Figura 44 y descritos a continuación:
    a. ***Application name***: es el nombre que aparecerá para referenciar a la aplicación en el dispositivo del usuario. Para las aplicaciones de este proyecto se establecen como HealthCoah y HealthCoach Admin.
    b. ***Project name***: es el nombre de la carpeta del proyecto en Eclipse.
    c. ***Package name***: representa el nombre asignado al paquete de la aplicación siguiendo las especificaciones de Java. En este proyecto se ha establecido como com.example.healthcoach.
    d. ***Minimum Required SDK***: establece la versión mínima de la API de Android que soporta la aplicación a desarrollar. Dicha versión debe encontrarse entre las disponibles en el SDK instalado en la máquina local. Establecer una versión baja asegura llegar a un mayor número de dispositivos pero limita las funciones disponibles. Estableciendo este parámetro a un nivel de API 8 aseguramos la compatibilidad con un 95% de dispositivos Android.
    e. ***Target SDK***: Establece la versión máxima de API compatible con nuestra aplicación.
    f. ***Compile With***: Representa la versión de API disponible en el SDK con la que se compilará el código fuente de la aplicación desarrollada. Normalmente será la versión

de API disponible más reciente o bien la primera versión que soporte todas las características que se deseen incluir.

g. ***Theme***: Establece el tema de la aplicación por defecto de entre una lista de opciones. Es una forma rápida de obtener, por ejemplo, una aplicación con tema de tonos claros u oscuros.

3. Al pulsar en *Next* se muestra una pantalla que ayuda a crear los iconos de la aplicación para las distintas densidades de pantalla.
4. Al pulsar de nuevo en Next se navega a la última pantalla en la que podemos definir el tipo y nombre de la actividad (*Activity*) principal.
5. Pulsar Finish para finalizar la creación del proyecto.

3.2.3.3 Estructura de proyecto Android

Después de crear un proyecto siguiendo los pasos definidos en el **Apartado 3.2.3.2,** puede observarse como se ha generado automáticamente una estructura de carpetas y ficheros como muestra la Figura 45.

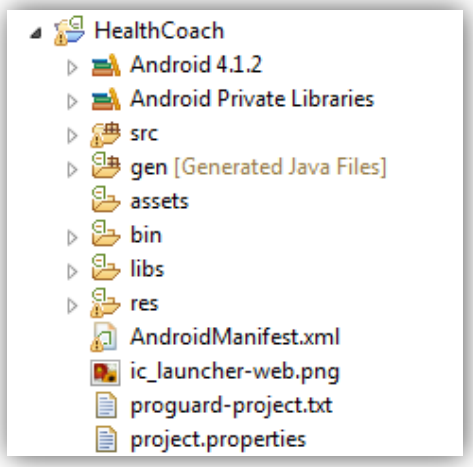


Figura 45. Estructura de proyecto Android en Eclipse IDE

La finalidad de cada uno de los elementos generados es:

Carpeta Android x.x.x

Alberga la API de Android correspondiente al número indicado en el nombre de la carpeta.

Carpeta Android Private Libraries

En su interior se guardan las librerías externas utilizadas en el proyecto.

Carpeta src

Contiene las clases Java generadas en el proyecto Android.

Carpeta gen

Contiene ficheros generados automáticamente al compilar el proyecto entre los que destaca el archivo ***R.java***, que contiene las referencias a todos los recursos de la aplicación. El contenido de este fichero no debe modificarse manualmente.

Carpeta bin

Carpeta con los binarios de la aplicación, es decir, las clases compiladas y el fichero .apk con el empaquetado final listo para ser instalado en un dispositivo.

Carpeta libs

Contiene las librerías externas con formato .jar utilizadas por la aplicación, como se observa en la Figura 46.

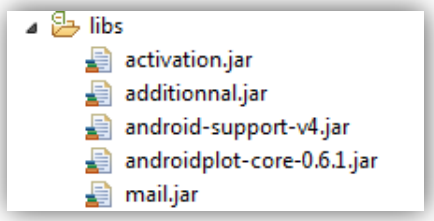


Figura 46. Librerías Java contenidas en la carpeta libs

Carpeta res

Alberga todos los recursos utilizados en el proyecto (Figura 47), como son *layouts*, imágenes, cadenas de texto, etc., mediante una clasificación en subcarpetas según el tipo de recurso:

- *drawable-[densidad]* – Existe una carpeta *drawable* por cada densidad de pantalla con imágenes de dicha densidad, con el propósito de obtener una calidad de imagen adecuada sin importar el tamaño o resolución de la pantalla.
- *layout* – Contiene los ficheros XML encargados de definir las interfaces o vistas asociadas a cada Activity.
- *values* – En su interior existen ficheros XML que definen ciertas etiquetas para su reutilización en los *layouts*, como el fichero *strings.xml* con la definición de los literales, es decir, las cadenas de texto usadas a lo largo de la aplicación. También contiene un fichero que define estilos de visualización denominado *styles.xml*, otro archivo llamado *color.xml* define etiquetas para los colores utilizados. Por último *dimens.xml* establece una serie de dimensiones para tamaño de fuentes, márgenes, etc.

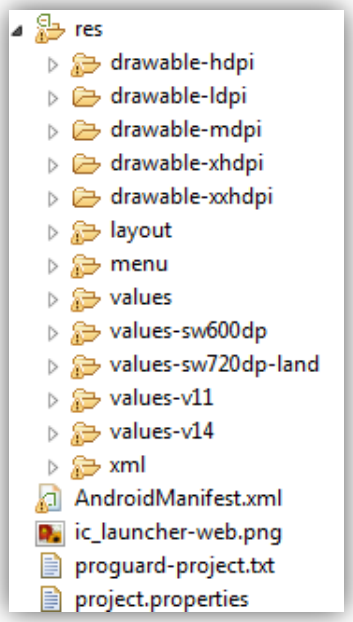


Figura 47. Estructura de recursos del proyecto dentro en la carpeta res

Cuando se realiza la compilación del proyecto, cada uno de los archivos contenidos en esta carpeta pasan a referenciarse en el fichero *R.java* anteriormente descrito.

Fichero AndroidManifest.xml

Contiene la declaración de todas los Activity utilizados en la aplicación, así como la gestión de permisos. En el **Apartado 4.3** se detalla su composición.

3.2.3.4 Proceso de compilación y ejecución de la aplicación

Por defecto, la compilación del proyecto en Eclipse se realiza de forma transparente en segundo plano cada vez que se crea o modifica un fichero de la aplicación. Si se dispone de un ordenador o portátil con recursos limitados quizás se desee deshabilitar, para ello desde el menú *Project* se desmarca la opción *Build Automatically*, como muestra la Figura 48.

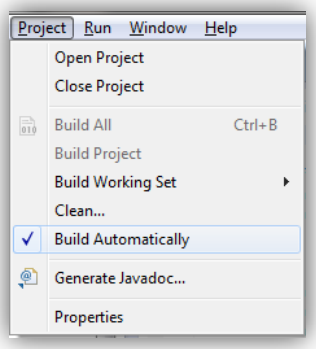


Figura 48. Gestión de la compilación automática

A partir de ese momento, deberá realizarse la compilación en modo manual, lo que resulta tan sencillo como abrir la carpeta del proyecto y pulsar en *Build Project*.

En cuanto a la ejecución de la aplicación, puede realizarse de dos formas:

- **Ejecución en AVD (Android Virtual Device).** Es el tipo de ejecución predefinida si no existe un dispositivo Android conectado al ordenador. ADT posibilita emular un dispositivo estableciendo una serie de características como arquitectura del procesador, tamaño y densidad de pantalla, memoria RAM, tamaño de almacenamiento externo, etc., como ilustra la Figura 49. El acceso a esa pantalla se realiza desde el menú de Eclipse, pulsando el botón  . Una vez creado el AVD, éste queda a disposición de futuras ejecuciones de proyectos Android cuando se selecciona en el menú de Eclipse en *Run* > *Run*.

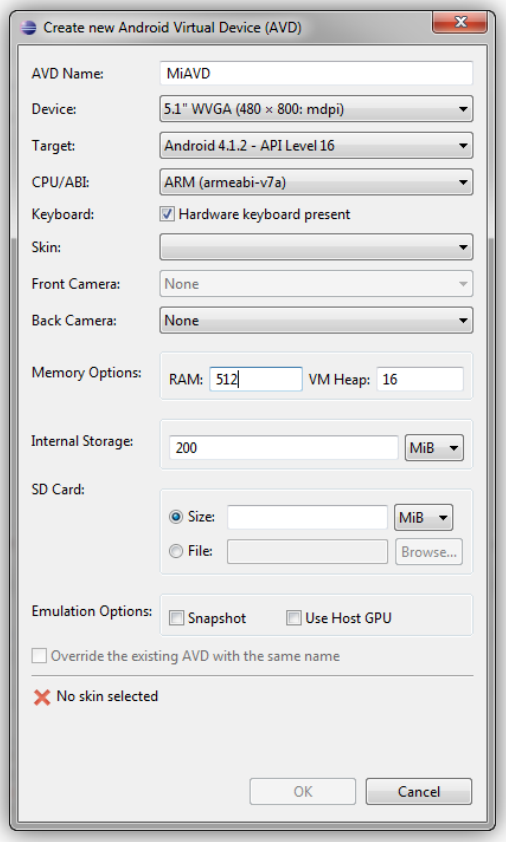


Figura 49. Pantalla de creación de un dispositivo virtual Android

- **Ejecución en dispositivo físico.** El plugin ADT permite ejecutar directamente la aplicación desarrollada en un dispositivo real, compilando el código y generando un .apk firmado que es transferido automáticamente al dispositivo real. En comparación con los AVD, existe una clara ventaja en cuanto a velocidad de carga de la aplicación y velocidad de interacción. Para utilizar esta función deben cumplirse estos requisitos:

- Disponer en el ordenador/portátil de los drivers del terminal Android. Si no se dispone de ellos o no se encuentran con facilidad, Google pone a disposición de los desarrolladores unos drivers universales en el enlace: http://developer.android.com/tools/extras/oem-usb.html.
- Debe habilitarse la depuración a través de USB en el terminal móvil, para ello, desde Ajustes > *Opciones de desarrollo* > *Depuración*, marcar la opción *Depuración USB.*
- Permitir la instalación de aplicaciones de origen desconocido desde *Ajustes* > *Seguridad* > *Administración de Dispositivos*, habilitar *Orígenes desconocidos*.
- Disponer de un cable USB para conectar el dispositivo móvil al ordenador.

Si se conecta un dispositivo físico al ordenador cumpliendo los requisitos anteriores, cuando se invoque a la ejecución del proyecto desde el menú de Eclipse en *Run* > *Run*, la aplicación aparecerá en el dispositivo en unos segundos. La primera vez que se realice esta tipo de ejecución con un terminal aparecerá la ventana mostrada en la Figura 50, siendo necesario seleccionar la referencia al dispositivo Android y pulsar en el botón *OK*.

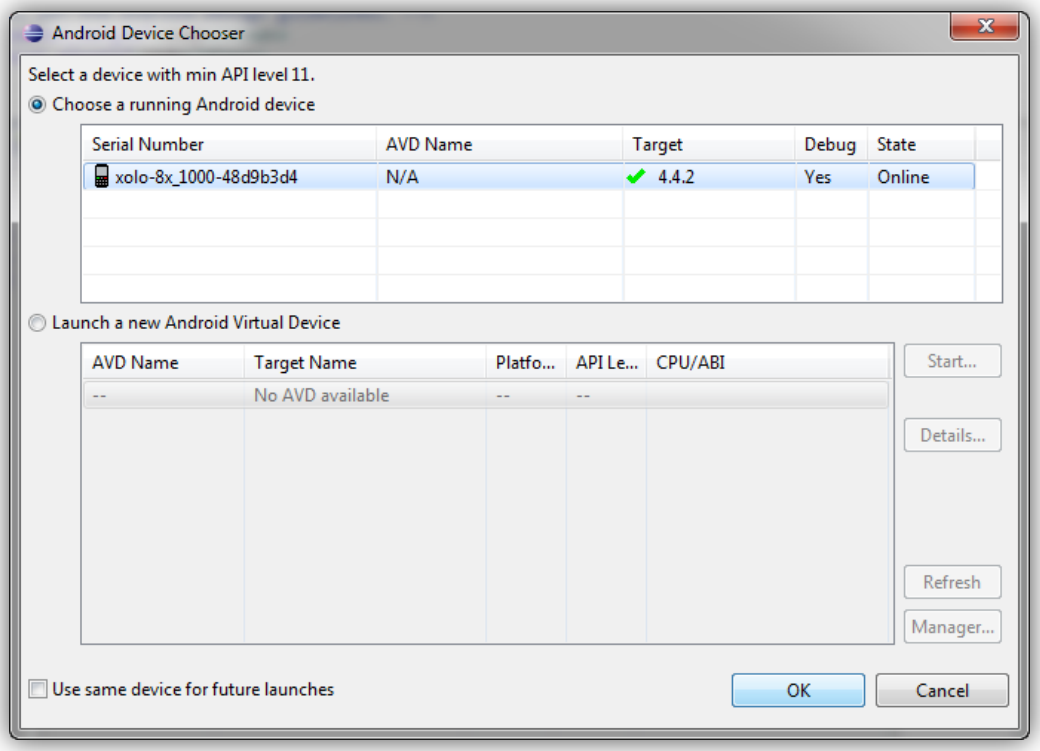


Figura 50. Detección en Eclipse de dispositivo físico Android

3.2.3.5 Depuración de código

ADT se integra perfectamente con la herramienta de depuración de Eclipse, para depurar un proyecto Android basta con ir al menú *Run* > *Debug as* y seleccionar *Android Application.*

Para abrir la vista de Eclipse optimizada para tareas de depuración, donde se establecen los puntos de ruptura, hay que navegar por el menú *Window* > *Open Perspective* y pulsar en *Debug*. En ese momento se visualizará el IDE con una interfaz parecida a la de la Figura 51.

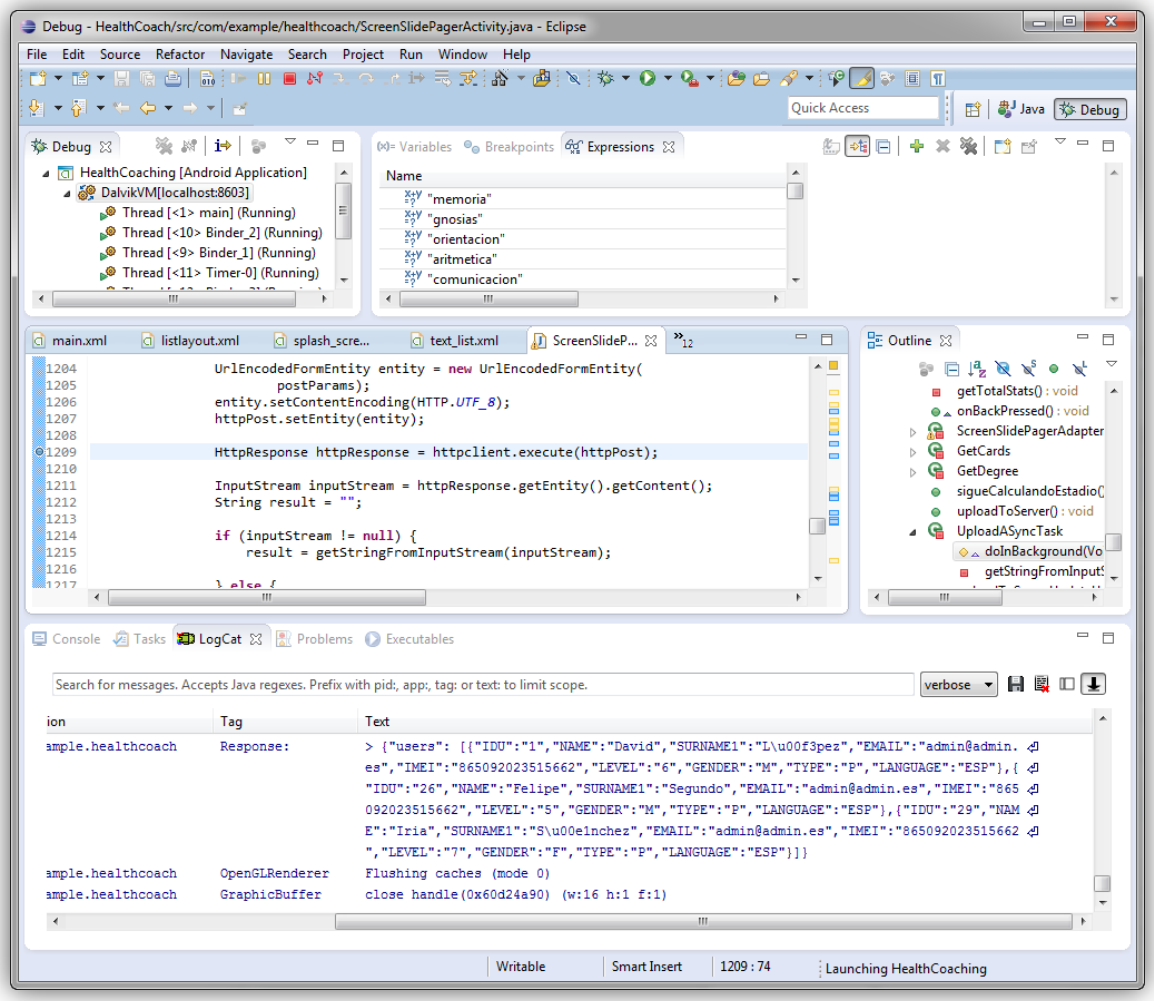


Figura 51. Perspectiva de depuración en IDE Eclipse

LOGCAT

Android utiliza el fichero LogCat para almacenar todos los eventos generados en el sistema. Para depurar y facilitar la localización de posibles errores, la clase Log puede escribir en el LogCat definiendo el nivel de trazabilidad:

- Log.e() – Errors (errores).
- Log.w() – Warnings (advertencias).
- Log.d() – Debugging (depuración).
- Log.i() – Information (información).

3.2.4 Toad for MySQL

La herramienta gratuita ***Toad for MySQL*** [78], desarrollada por *Quest Sofware* aunque posteriormente adquirida por Dell en 2012, permite desarrollar código SQL, crear y ejecutar consultas y administrar bases de datos MySQL. También facilita la transferencia de información mediante la exportación/importación de datos.

Puede descargarse desde la siguiente url: https://www.toadworld.com/m/freeware/1469. Una vez instalada la herramienta, la primera pantalla solicitará información para realizar la conexión con la instancia MySQL a administrar. La Figura 52 muestra un ejemplo de conexión utilizando los parámetros reales de la base de datos utilizada en la realización de este Proyecto. Como puede observarse es necesario indicar el tipo de conexión, el host o dirección del servidor, usuario y contraseña, nombre de la base de datos y el puerto. Una vez establecidos pueden guardarse asociando un nombre de conexión para realizar posteriores conexiones de forma más rápida.

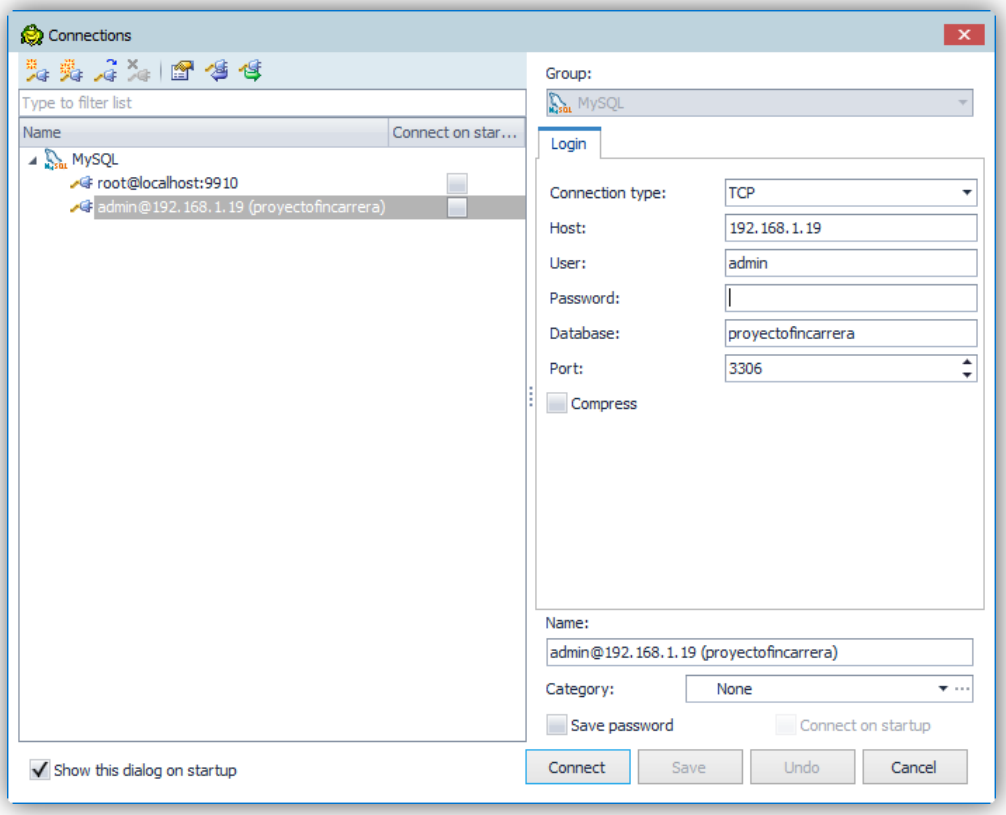


Figura 52. Datos de conexión a la base de datos proyectofincarrera desde Toad for MySQL

3.2.5 Postman

La herramienta Postman [79], desarrollada como un plugin gratuito del navegador Google Chrome, facilita explorar de forma rápida y fácil las peticiones HTTP y llamadas-respuestas a APIs remotas sin necesidad de escribir ningún código adicional. Para ello cuenta con soporte para las principales operaciones HTTP: GET, POST, PUT, DELETE Y HEAD.

Permite además añadir parámetros, cabeceras HTTP o cuerpos XML/JSON en la petición o *request*, y dispone de un visor de formatos JSON, XML y HTML para formatear correctamente las respuestas recibidas.

Existen otros plugins para Chrome con funcionalidad similar a Postman, como son JaSON o Advanced REST client.

Para el desarrollo de este proyecto se ha aprovechado esta herramienta para realizar pruebas rápidas de la API REST desarrollada en PHP, que permite acceder a la base de datos MySQL. De esa manera podemos simular una llamada a la API como si se realizara desde la aplicación cliente Android. La Figura 53 muestra una petición GET a la url con formato REST:

```
192.168.1.19:8080/api/cards/1/2/ESP
```

La url establece la dirección del servidor que provee la API, la carpeta donde se encuentra (/api), la acción a realizar que es devolver las tarjetas (*CARDS*) y un conjunto de parámetros que en este caso representan por orden el identificador de usuario, nivel de tarjetas e idioma de las mismas. En la Figura 53 se muestra el resultado de la invocación GET desde la interfaz de Postman.

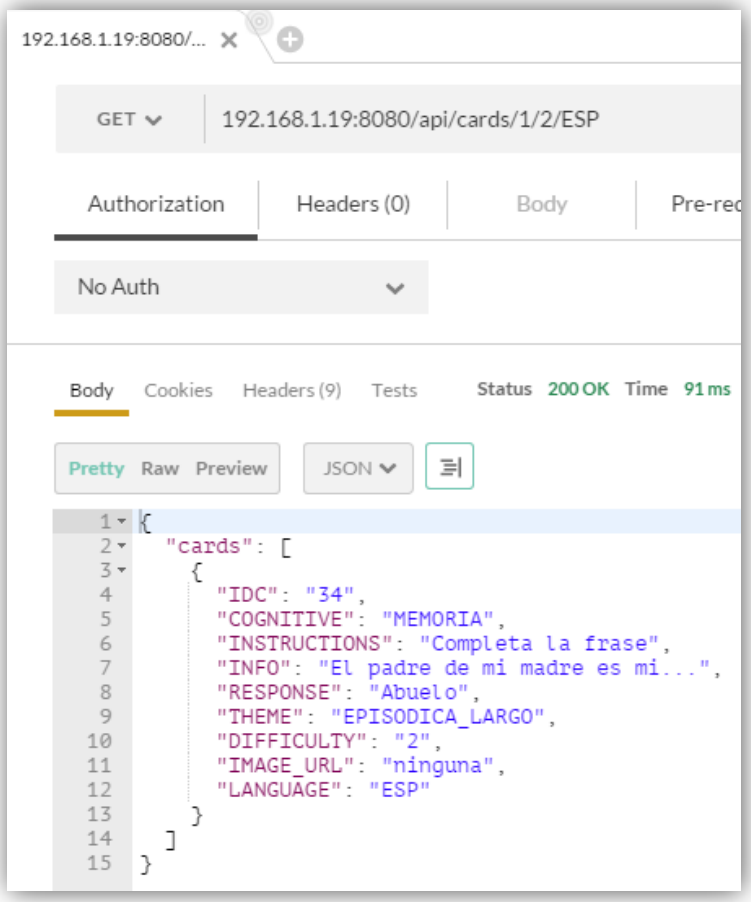


Figura 53. Petición GET al servicio REST remoto desde herramienta Postman

3.2.6 Putty

El cliente de acceso Putty [80] permite establecer conexiones remotas mediante SSH, Telnet o RLogin con servidores que se encuentren en la misma red o en el exterior. Posee una licencia de software libre y puede instalarse en plataformas Windows y Linux. La dirección de descarga es: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

Putty permite guardar datos de sesiones anteriores incluidas sus preferencias, como puede comprobarse en el ejemplo de la Figura 54, donde puede encontrarse una entrada dentro de *Saved Sessions* con el nombre RaspberryPi. Una vez seleccionada y pulsando el botón Load cargará los datos previamente guardados como son el *host name* que es el nombre o dirección IP del servidor y el port o puerto asociado al *connection type* (tipo de conexión). En el caso mostrado la conexión se realiza por SSH que corresponde al puerto reservado 22.

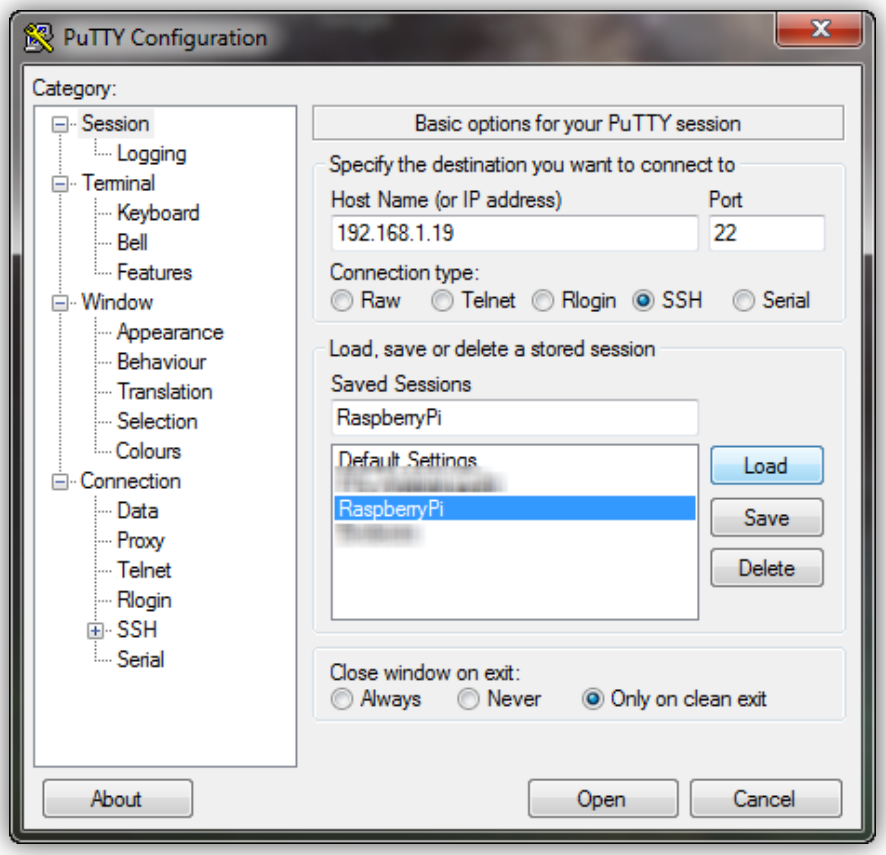


Figura 54. Pantalla de opciones de conexión en cliente SSH Putty

Esta herramienta ha facilitado dentro del desarrollo del sistema HealthCoach, acceder a una consola del servidor remoto con sistema operativo Raspbian para poder gestionar todas las herramientas que aloja, como la API REST, la base de datos MySQL, la instancia de Apache, etc. donde se aloja la API REST de acceso a base de datos.

3.2.7 Filezilla

Filezilla [81] es un famoso cliente FTP (File Transfer Protocol) de código abierto y gratuito, disponible para las plataformas Windows, Linux, Mac OS, BSD y otras.

Este cliente soporta además los protocolos FTPS (FTP sobre SSL/TLS) y SFTP (SSH FTP), realizar transferencias de ficheros superiores a los 4GB de tamaño, ajuste de velocidades máximas de transferencia, edición y búsqueda remota de ficheros, doble navegador de archivos para visualizar la máquina local y remota, etc.

En la realización del proyecto HealthCoach se ha utilizado para gestionar la transferencia de ficheros PHP alojados en el servidor Apache del dispositivo RaspberryPi. La Figura 55 muestra un ejemplo de conexión con el servidor remoto, siendo necesario indicar la dirección del servidor, el protocolo de conexión, nombre de usuario y contraseña.

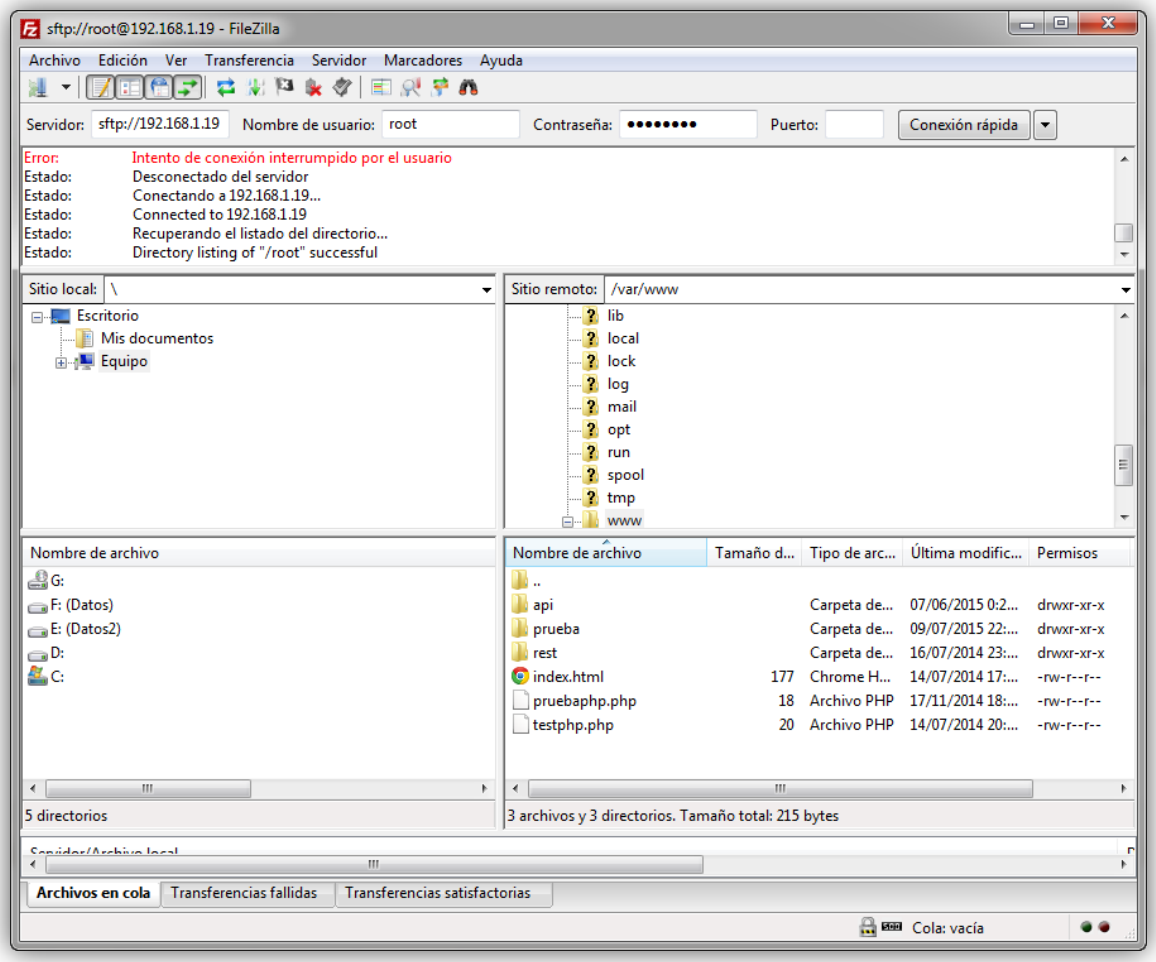


Figura 55. Conexión SFTP con el servidor remoto Raspberry Pi

Capítulo 4

Arquitectura de Android

Android establece su arquitectura en cuatro capas diferenciadas (Figura 56) [82] que proveen servicios a las capas inmediatamente superiores y a su vez los consumen de las inmediatamente inferiores. Todos los componentes que conforman las capas están basados en el software libre.

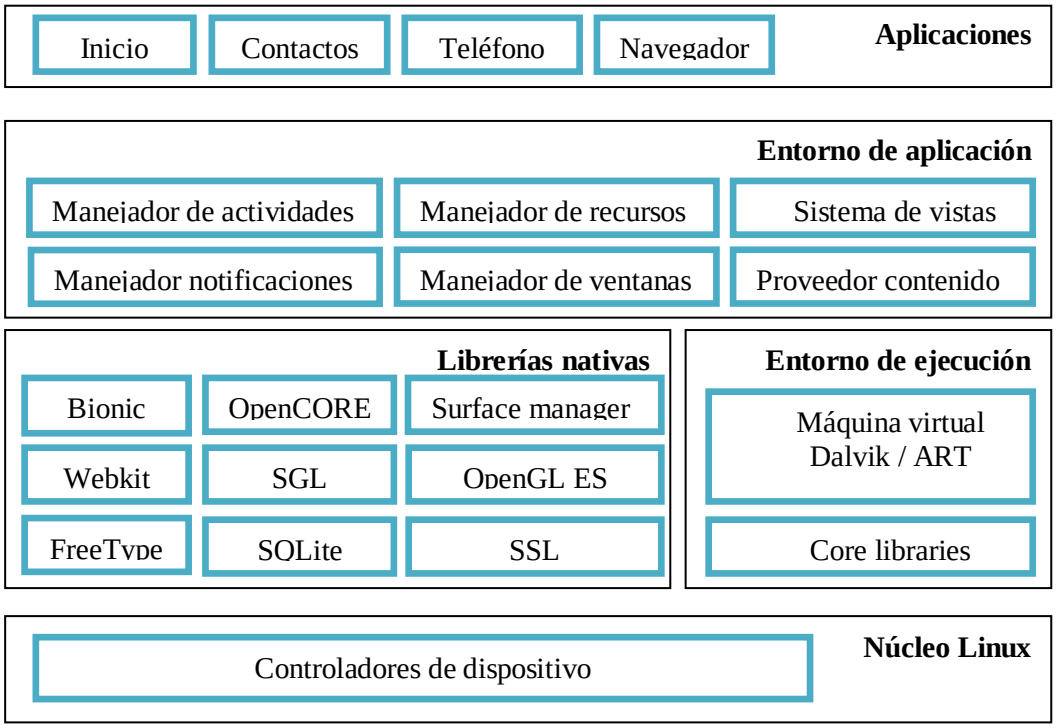


Figura 56. Capas que conforman la arquitectura de Android

A continuación se describen cada una de las capas, comenzando desde el núcleo hasta los niveles de alto nivel:

- **Núcleo de Linux**: El núcleo de Android está basado a su vez en el núcleo de Linux, utilizando en sus comienzos la versión 2.6.25, pero relevada por la 3.4 o 3.10 en las últimas versiones de Android. Por tanto, comparte con Linux servicios que establecen la base del sistema, como son la gestión de la memoria y los procesos, modelos de comunicación de red y seguridad, gestión de controladores del hardware del dispositivo móvil, así como el resto de funciones típicas del núcleo de un sistema operativo.

- **Librerías nativas:** La segunda capa de Android incluye un conjunto de librerías escritas en lenguaje C o C++ y compiladas en código nativo del microprocesador, a disposición de los componentes característicos del sistema operativo. Los desarrolladores también pueden hacer uso de estas librerías a través del entorno de aplicación que compone la tercera capa de la arquitectura. Entre las más importantes se encuentran:

- **Bionic**: basada en la librería estándar de C (libc), es adaptada y optimizada por Google para permitir su funcionamiento en procesadores con baja frecuencia.
- **OpenCORE**: librería que permite la reproducción y grabación de diversos contenidos multimedia (audio, vídeo e imágenes).
- **Surface Manager**: Establece el sistema de ventanas definiendo el flujo de visualización y navegación entre ellas. mediante presentación 2D y 3D.
- **Webkit**: motor de navegación web para representar contenido HTML utilizado por el navegador nativo de Android y las vistas WebView. Esta librería supone también la base de los navegadores *Google Chrome* y *Safari* de Apple.
- **SGL:** provee un motor de gráficos 2D a disposición de las aplicaciones.
- **OpenGL ES:** versión de OpenGL para sistemas embebidos, provee una API para gráficos 3D.
- **FreeType:** gestor de tipos de fuentes del sistema.
- **SQLite:** motor de bases de datos relacionales a disposición de las aplicaciones en Android.
- **SSL**: librería para securizar comunicaciones mediante servicios de encriptación denominados *Secure Socket Layer*.

- **Entorno de ejecución:** Junto con las librerías nativas, el entorno de ejecución de Android completa la segunda capa de la arquitectura de Android. Se compone de las *core libraries* y de la máquina virtual Dalvik.

  - **Core libraries:** Proporcionan parte de la funcionalidad presente en las librerías de Java SE y Java ME.
  - **Máquina virtual Dalvik / ART:** El entorno de ejecución de Android se basa en una versión reducida de la máquina virtual de Java (debido a las limitaciones de procesador y memoria de los dispositivos móviles) que Google denominó ***Dalvik***. Como peculiaridad, no trabaja directamente con ficheros .class sino con una variación de éstos denominados .dex, que se generan en el tiempo de compilación y proveen un mayor rendimiento en dispositivos de rendimiento limitado. Dalvik permite múltiples instancias de máquinas virtuales de forma simultánea y gestiona la memoria y el soporte multihilo. Google ha introducido una nueva máquina virtual denominada ***ART*** (Android Runtime) para las últimas versiones de Android, siendo la predefinida en Lollipop. Introduce importantes ventajas en la compilación y en el recolector de basura para incrementar el rendimiento de las aplicaciones.

- **Entorno de aplicación:** Son los servicios o APIs escritos enteramente en lenguaje Java que se encuentran a disposición de los desarrolladores para utilizarlos en sus aplicaciones. Su uso simplifica el desarrollo a la vez que enriquece las funcionalidades al reutilizarse el código contenido por dichos servicios. Entre los más importantes se encuentran:

  - **Sistema de vistas**: del inglés *View System*, representa la implementación de la interfaz gráfica de usuario (GUI) mediante el uso de componentes como botones, listas, etiquetas, spinner (combo box), etc., que conforman cada una de las vistas de las aplicaciones.

- **Proveedor de contenido**: del inglés *Content Provider*, permite compartir datos de forma encapsulada entre aplicaciones, como es el acceso a los datos de la agenda del teléfono, a los mensajes SMS, etc. Se explicará en mayor detalle en el **Apartado 4.1.3**.
- **Manejador de actividades**: del inglés *Activity Manager*, se encarga de gestionar el ciclo de vida de las actividades que conforman la aplicación.
- **Manejador de notificaciones**: del inglés *Notification Manager*, es el mecanismo mediante el cual Android informa al usuario de los eventos generados por las aplicaciones, como puede ser una llamada perdida, la actualización de una aplicación, la llegada a una posición GPS concreta, etc. Todas las notificaciones se gestionan a través de este manejador para mostrarse de una forma homogénea.
- **Manejador telefónico**: del inglés *Telephony Manager*, gestiona todas las funciones habituales de un teléfono como las llamadas de voz, mensajería SMS o MMS, etc.
- **Manejador de recursos**: del inglés *Resource Manager*, permite gestionar los elementos de la aplicación externos del código, como son las imágenes, sonidos, *layouts*, cadenas de texto asociadas a idioma, etc.
- **Manejador de localización**: del inglés *Location Manager*, permite acceder al sistema de posicionamiento geográfico del terminal por medio del GPS o de las redes telefónicas.
- **Manejador de ventanas**: del inglés *Window Manager*, organiza las ventanas que se mostrarán en la pantalla y que serán posteriormente completadas por las actividades.

- **Aplicaciones**: Representan la capa superior de la arquitectura Android, compuesta tanto por las aplicaciones preinstaladas y proporcionadas por Android como las desarrolladas por terceros. En esta capa se encuentra la aplicación principal del sistema denominada lanzador (del inglés launcher) encargada de mostrar los escritorios y lanzar las aplicaciones que contienen. Como puntos comunes, todas deben ejecutarse en la máquina virtual de Android, ya sea Dalvik o ART, y haber sido desarrolladas en Java mediante el SDK Android proporcionado por Google o bien en C/C++ mediante el Native Development Kit (NDK) de Android.

4.1 Componentes de una aplicación

Las aplicaciones Android están formadas por cinco tipos de componentes que deben ser registrados en el archivo *AndroidManifest.xml*. Toda aplicación posee al menos uno de ellos, pero usualmente están formadas por una combinación de estos componentes.

Es importante remarcar que todos los componentes descritos a continuación son ejecutados en el hilo principal de la aplicación, por lo que no deberían desarrollar tareas pesadas que bloqueen su ejecución. Si fuera necesario realizar una tarea de estas características habría que recurrir a nuevos hilos de ejecución, la clase *AsyncTask* de Android facilita la creación de tareas asíncronas mediante hilos paralelos al principal de la aplicación.

4.1.1 Activity

En Android, cada una de las pantallas que componen la aplicación se representan por medio de una actividad o *Activity*. Su finalidad es la de crear la interfaz de usuario en conjunción con las vistas (representadas por la clase *View*), por lo que típicamente son necesarias varias actividades para completar la experiencia de usuario. Por ejemplo, en una aplicación de búsqueda de información de películas, una actividad representaría la pantalla con el campo para introducir el nombre de la película a buscar, otra mostraría la información encontrada como el director o los actores, otra más mostraría un listado de películas relacionadas, etc. Toda actividad definida en la aplicación debe heredar de la clase *Activity*.

4.1.1.1 Ciclo de vida de una aplicación

El ciclo de vida de una actividad y en extensión el de una aplicación, es gestionado directamente por el sistema Android mediante una pila que contiene las actividades previamente mostradas, de esa forma es posible retroceder a una actividad anterior ya sea programáticamente o mediante el uso del botón *atrás*.

Los estados en los que puede encontrarse una actividad pueden visualizar en la Figura 57 [82]. A continuación, se describe el significado de cada uno de ellos:

- **Activa**: del inglés *Running*, indica que la actividad se encuentra en lo alto de la pila por lo que tiene el foco de ejecución.
- **Visible**: del inglés *Paused*, indica un estado en el que la actividad se encuentra visible pero no tiene el foco, por ejemplo al mostrarse parcialmente en segundo plano.
- **Parada**: del inglés *Stopped*, se alcanza este estado cuando no es visible la actividad.
- **Destruida**: del inglés *Destroyed*, significa que la actividad es eliminada por completo de la pila por el sistema Android o mediante la invocación del método finish().

Al pasar de un estado a otro, se generan una serie de eventos (visibles en la Figura 57) que pueden ser capturados por métodos de la clase Activity:

- **onCreate()**: representa la creación de la actividad.
- **onStart()**: define el momento en que la actividad pasa a ser visible.
- **onResume()**: al invocar el método, se establece el foco en la actividad que ya era visible previamente.
- **onPause()**: con su invocación, la actividad pierde el foco pero sigue siendo visible.
- **onStop()**: captura el momento en que la actividad deja de ser visible pero no llega a ser destruida. Si el sistema requiere liberar memoria, es posible que destruya la actividad sin pasar por este estado.
- **onRestart()**: la actividad vuelve a ser visible después de haber pasado por un onStop().
- **onDestroy()**: método invocado justo antes de proceder a la destrucción de la aplicación.

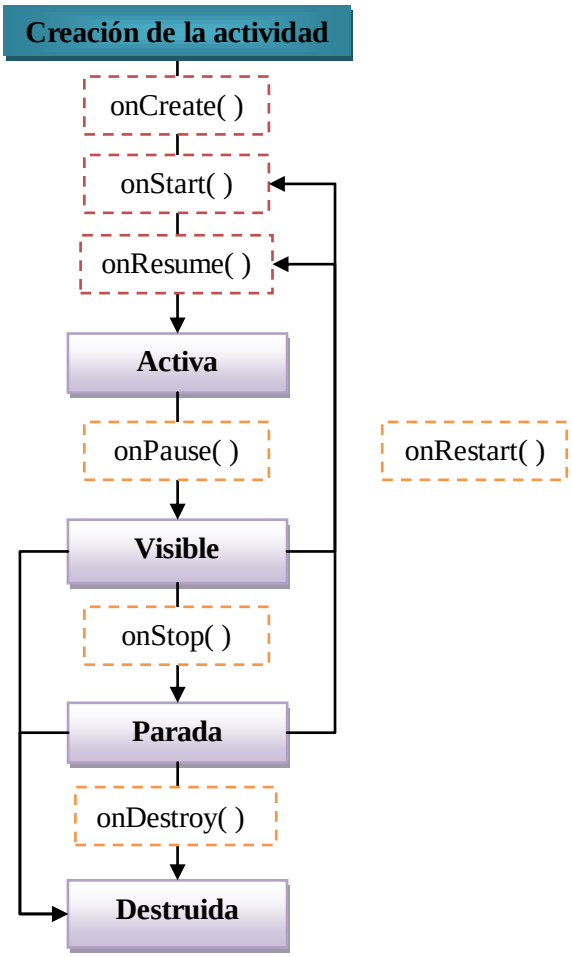


Figura 57. Ciclo de vida de una actividad en Android

Toda aplicación lanzada dentro del sistema puede ser destruida dentro del programa que la define (programáticamente) mediante el método *finish()* o bien por el propio sistema Android en casos críticos que requieren liberar memoria. La decisión de Android para finalizar uno u otro proceso se basa en la importancia de la aplicación para el usuario, determinada por las aplicaciones y servicios que ejecutan y en qué estado se encuentran. Una vez finalizada la aplicación, se pierde también el estado que tenía justo antes de ese evento, por lo que es responsabilidad del programador almacenar el contexto de la aplicación, por ejemplo, a través del método *onSaveInstanceState(Bundle)*.

4.1.2 Service

Los servicios son componentes de una aplicación que se ejecutan en segundo plano y no requieren de interacción con el usuario. Normalmente, suelen ser procesos de larga duración que han sido lanzados desde una aplicación y permanecen en *background* aunque cambie el foco a otra actividad, o incluso aunque la actividad que lo generó sea destruida.

Como ejemplo de servicio podemos nombrar el caso de un cronómetro, en el que la actividad principal se encarga de representar la interfaz del contador numérico así como de los botones que lo

ponen en marcha o paran. En el momento de pulsar el botón de inicio, se lanza un servicio en segundo plano encargado de contabilizar el tiempo y representarlo en pantalla, de tal forma que si salimos de la aplicación el proceso sigue corriendo y no se detiene el cronómetro. Al recuperar el foco de la aplicación la cuenta del tiempo se mantendría intacta hasta que se pulsara el botón encargado de detenerlo.

Podemos encontrar dos tipos de servicios, diferenciados por su propósito y tipo de inicialización:

- **Started**: Son los iniciados con el método *startService()*. Una vez arrancados se ejecutan en segundo plano de forma indefinida hasta que se, incluso aunque la actividad desde la que se crearon fuera destruida. Para pararlos es necesario utilizar el método *onDestroy()*.
- **Bound**: Iniciados con *bindService()*, son servicios que permiten interactuar con ellos, bien desde la actividad que los lanzó o desde cualquier otra aplicación. Posibilitan el envío de solicitudes, facilitan resultados, etc., mediante los métodos ofrecidos por su instancia.

4.1.2.1 Ciclo de vida de un servicio

Como acabamos de ver, existen dos tipos de servicios con funciones diferenciadas, al igual que su ciclo de vida, como puede observarse en la Figura 58.

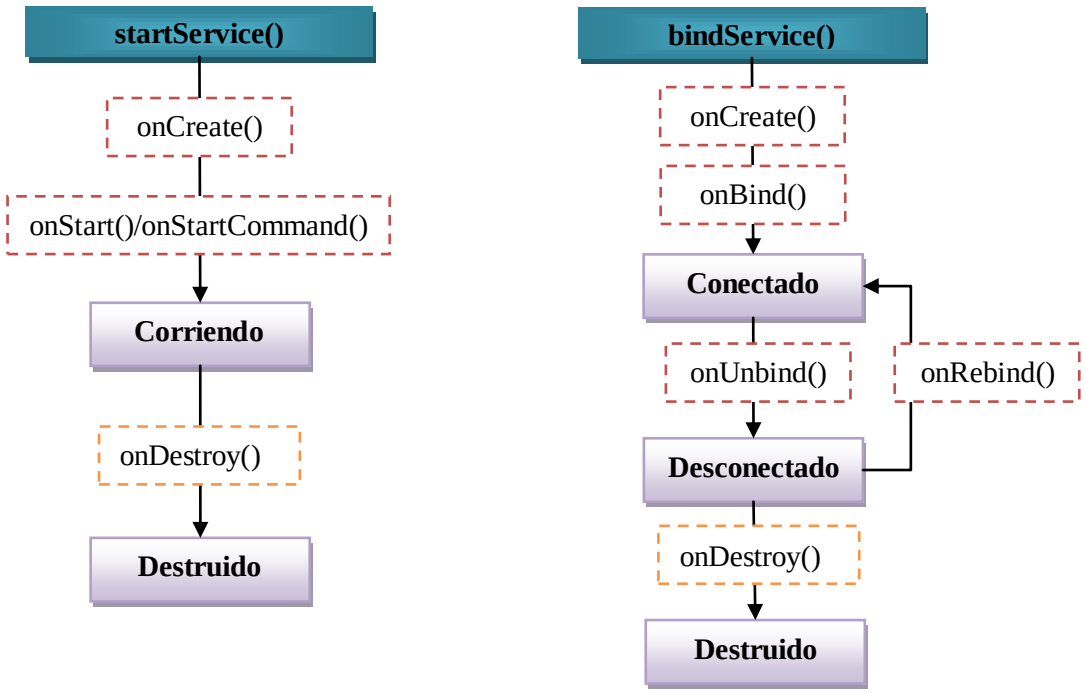


Figura 58. Ciclos de vida de los servicios Android

Existen un par de métodos comunes en ambos tipos de servicios, uno es *onCreate()* que es lanzado al instanciar el servicio y permite la configuración del mismo. El otro es *onDestroy()*, lanzado por el sistema, permite guardar los datos y liberar recursos antes de la destrucción definitiva del servicio.

En el arranque de los servicios ***started*** se invoca al método onStartCommand(). A partir de ese momento, es necesario utilizar el método stopService() desde la actividad que lo invocó o el método stopSelf() desde el propio servicio para conseguir detenerlo.

Para los servicios ***bound*** el arranque lanza automáticamente el método *onBind()* desde donde se conecta el servicio con la actividad que lo invoca. A partir de ese momento el servicio estará disponible para interactuar con él, siendo necesario utilizar el método *unbindService()* para desconectar de la actividad o componente asociado generando una llamada a *onUnbind()*. Una vez desconectado pero sin llegar a ser destruido, puede realizarse una reconexión que invocará al método *onRebind()*.

4.1.3 Content provider

Los proveedores de contenido (*Content Provider*) permiten abstraer colecciones de datos en servicios con el propósito de consumir información de otras aplicaciones, o bien poner a disposición del resto las generadas en nuestra propia aplicación. En resumen, permiten el intercambio de datos entre aplicaciones Android, ya que por defecto, las bases de datos son de acceso privado a la propia aplicación que las creó. Generalmente, se utiliza el motor de base de datos SQLite como capa de persistencia para guardar los datos que gestionará el proveedor de contenido, pero puede utilizarse un fichero o cualquier otro formato disponible. La figura 59 muestra de forma esquematizada el funcionamiento de un Content provider.

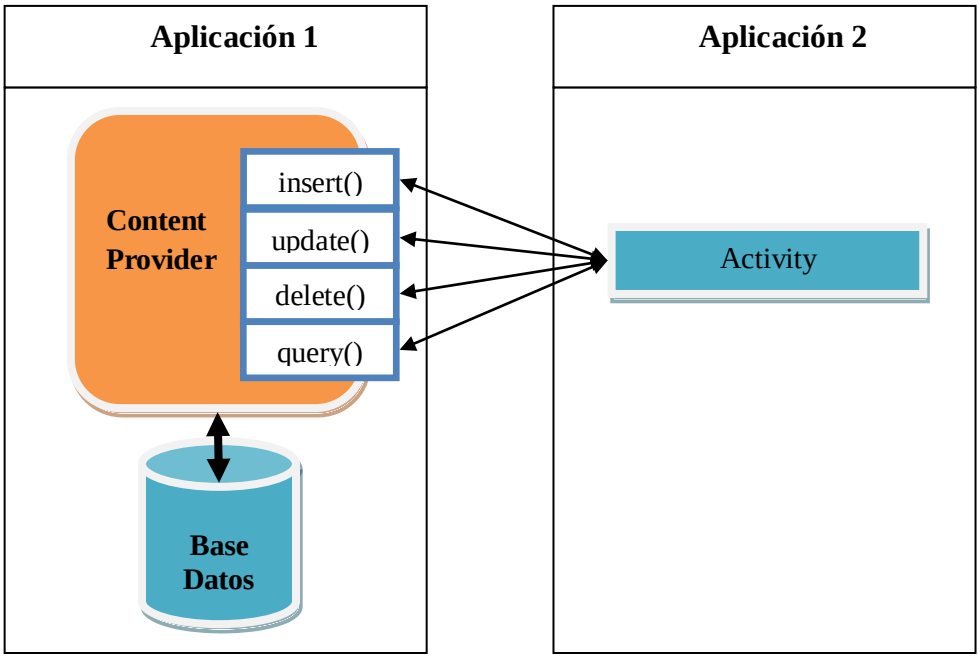


Figura 59. Funcionamiento de los proveedores de contenido en Android

Para acceder a la información alojada en un *ContentProvider* es necesario realizar una consulta que devolverá un objeto de tipo *Cursor* con una estructura similar a la de la Tabla 14, que contiene la estructura de datos correspondiente al ContentProvider con el registro de las llamadas del dispositivo, denominado *CallLog*.

_id	Date	Number	Duration	Type
1	10/05/2015 16:30	654656565	86	INCOMING_TYPE
2	10/05/2015 18:24	687253614	27	MISSED_TYPE
3	12/05/2015 09:22	691125678	268	OUTGOING_TYPE

Tabla 14. Estructura de datos del ContentProvider CallLog

Los proveedores de contenido son identificados a través de una URI, es decir una cadena de texto que representa un recurso informativo. El estándar RFC 2396 [83] define el formato de una URI de la siguiente forma:

```
<standard_prefix>://<authority>/<data_path>/<id>
```

Para los ContentProvider se establece el <standard_prefix> como *content*, por lo que utilizando el ejemplo anterior de proveedor de contenido para el registro de llamadas (CallLog) se utilizaría la siguiente URI para acceder a la información:

```
content://call_log/calls
```

Existen otros ContentProvider considerados de importancia en el sistema Android, como son la información relativa al historial y favoritos del navegador web mediante el elemento *Browser*, los contactos de usuario mediante *Contacts*, ficheros multimedia mediante *MediaStore*, preferencias del sistema mediante *Setting*, mensajería SMS y MMS mediante *Telephony*, eventos del calendario mediante *Calendar*, etc.

4.1.4 Broadcast receiver

Los *BroadcastReceiver* o receptores de anuncios, son componentes destinados en exclusiva a escuchar y capturar anuncios de tipo global. En su mayoría generados por el propio sistema Android pero también pueden lanzarse desde las aplicaciones. Ejemplos de anuncios Android son los generados por llamadas entrantes, batería baja, etc. El registro de un receptor de anuncios debe realizarse desde el fichero AndroidManifest.xml o bien con el método *registerReceiver()*.

A pesar de no contar con una interfaz, generalmente lanzan una actividad para informar del evento capturado o incluso hacer uso del *NotificationManager* para mostrar una notificación en la barra de estado del dispositivo.

No es necesario que la aplicación que creó el *BroadcastReceiver* se encuentre activa para poder capturar el anuncio registrado, el sistema se encargará de ponerla en marcha si fuera necesario.

4.1.4.1 Ciclo de vida de receptor un de anuncios

Como puede observarse en la Figura 60, el ciclo de vida de un receptor de anuncios es muy básico, únicamente dispone del método *onReceive()* que es lanzado en el momento de capturar el anuncio para poder realizar las acciones oportunas. Una vez se acaba de ejecutar el código de ese método, el objeto BroadcastReceiver es destruido.

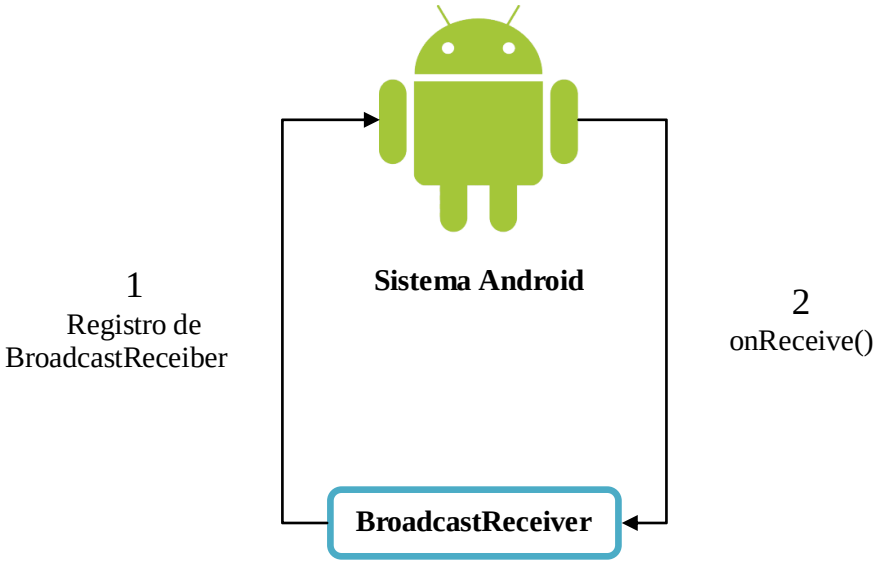


Figura 60. Ciclo de vida de un receptor de anuncios en Android

4.1.5 Intent

Los *Intent* son utilizados para lanzar otros componentes, representan la intención de realizar una acción. Pueden iniciar una *Activity* mediante el método *startActivity*, un *BroadcastReceiver* mediante *broadcastIntent*, arrancar un Service gracias a *startService* o comunicarse con él a través de *bindService*. Es decir, son capaces de lanzar otras actividades pero también de generar eventos que pueden ser capturados desde otra aplicación.

Pueden encontrarse dos tipos de Intent:

- **Intent explícito**: especifican el componente invocado gracias a *setComponent(ComponentName)* o *setClass(Context, Class)*.
- **Intent implícito**: no especifican un componente para realizar la acción, en su lugar, deben proveer suficiente información para que el sistema pueda decidir cuál de los componentes disponibles se adapta mejor al Intent ejecutado.

4.2 Interfaz de usuario

En Android, la interfaz gráfica que permite la interacción del usuario con la aplicación se define a través de ***layouts***. Éstos representan el aspecto visual de una *Activity*, estableciendo la estructura de la interfaz que contiene los elementos mostrados al usuario.

Aunque la forma habitual de crear un *layout* es declarando los elementos de las clases de tipo ***View*** en un archivo XML alojado en la carpeta ***/res/layout***, también se puede realizar mediante código alojado en el propio *Activity*, aunque esta forma es más laboriosa y menos eficiente.

Otra característica importante del layout, es controlar la disposición de los elementos en pantalla, definiéndose seis tipos en función de su estructura:

- **LinearLayout**: posiciona los objetos formando una fila o columna.
- **RelativeLayout**: distribuye los elementos en relación a otro o al padre.
- **TableLayout**: establece los elementos en filas y columnas a modo de tabla.
- **FrameLayout**: muestra un único elemento que podrá ser modificado de forma dinámica.
- **AbsoluteLayout**: posiciona los objetos en cualquier lugar de la pantalla utilizando coordenadas "x" e "y".

4.3 Android Manifest

Al margen de los componentes anteriormente analizados, existe un elemento imprescindible dentro de cualquier aplicación Android denominado AndroidManifest.xml, el cuál debe establecerse en la raíz del proyecto de forma obligatoria. Se trata de un manifiesto en formato XML con las siguientes propiedades [84]:

- Establece el **paquete Java** de la aplicación, que deber ser único.
- Especifica todas las actividades, proveedores de contenido, servicios o receptores de anuncios que posee la aplicación, estableciendo el nombre de las clases que implementan esos **componentes** y publicando sus capacidades.
- Describe los **permisos** necesarios para el correcto funcionamiento de la aplicación, tanto para acceder a funcionalidades o componentes externos como para permitir el acceso desde el exterior a componentes de nuestra aplicación. Ejemplos de permisos son acceso a Internet, a los contactos del dispositivo, etc.
- Define el **nivel** mínimo de **API Android** que la aplicación requiere para funcionar correctamente.

En la Figura 61 puede observarse un ejemplo de fichero AndroidManifest.xml donde se aprecian cada una de las propiedades anteriores.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.david.respuestagif"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="10"
        android:targetSdkVersion="21" />

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
              android:theme="@android:style/Theme.NoTitleBar.Fullscreen">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Figura 61. Ejemplo de AndroidManifest.xml

Capítulo 5

Análisis y Diseño

En este capítulo se analizan los requerimientos de usuario que surgen a partir del escenario propuesto, para posteriormente proceder a realizar un diseño del sistema que cumpla los requisitos y resuelva la problemática presentada en la fase de análisis.

5.1 Introducción a HealthCoach

El sistema software desarrollado en este PFC, designado **HealthCoach**, nace con el objetivo de crear una aplicación estándar capaz de adaptarse fácilmente al tratamiento de diversas dolencias médicas. En esta memoria, se describe un ejemplo práctico con ejercicios para potenciar y mantener las capacidades cognitivas de los enfermos de Alzheimer y ayudar a los terapeutas/familiares en el seguimiento de la enfermedad mediante la monitorización y diagnóstico del enfermo.

A su vez se pretende que el desarrollo de **HealthCoach** ilustre de forma práctica las características de Android y sirva de ejemplo en el desarrollo de otras aplicaciones. Algunos de los aspectos más importantes son:

- Uso de componentes *Intent* y *Activity*.
- Interfaces de usuario mediante código Java o XML.
- Uso de tareas en segundo plano mediante *AsyncTask*.
- Uso de representación gráfica mediante *Fragments*.
- Acceso a información identificativa del dispositivo: IMEI.
- Comunicaciones mediante HTTP.
- Recuperar información de JSON.
- Uso de recursos externos: imágenes.
- Gestión del fichero AndroidManifest.xml.
- Representación de gráficas estadísticas.
- Envío de correo electrónico.
- Acceso al estado de conexión Wi-Fi y datos móviles.

5.1.1 Escenario propuesto

Una vez realizada la introducción del sistema, conviene definir los escenarios concretos de uso de la aplicación para así facilitar la toma de especificaciones o requisitos de usuario.

En primer lugar se tendrá en cuenta un ***entorno doméstico*** en el que existen uno o varios enfermos de Alzheimer que utilizarán la aplicación de pacientes para realizar ejercicios de terapia cognitiva, así

como un familiar, amigo o cualquier persona encargada de gestionar los avances realizados a través de la aplicación de responsables.

El segundo escenario contemplará una ***institución médica***, residencia de ancianos o cualquier lugar dedicado al tratamiento en grupo de pacientes de Alzheimer, en el que pueden existir uno o varios dispositivos móviles con la aplicación de pacientes instalada, además de uno o varios terminales destinados al personal médico, terapeutas o cuidadores en los que realizar un seguimiento del progreso de los enfermos a través de la aplicación de responsables.

5.2 Requisitos de usuario

Para lograr los objetivos expuestos en el **Apartado 5.1** de este capítulo, deben establecerse unos requisitos de usuario mínimos para el sistema que va a desarrollarse. A continuación se recopilan las funcionalidades que HealthCoach debe cumplir:

- Registro en la aplicación.
- Listar usuarios disponibles en el dispositivo.
- Identificación en la aplicación.
- Envío automático de notificaciones por eventos en la terapia de paciente por medio de correo electrónico.
- Cálculo y visualización del grado de Alzheimer del paciente.
- Visualización de tarjetas cognitivas.
- Reconocimiento del habla del paciente.
- Adaptabilidad en los tipos de ámbitos cognitivos de las tarjetas presentadas al paciente.
- Generación de respuestas de voz por el sistema.
- Visualización de un listado de gráficas estadísticas de paciente.
- Visualización del detalle de cada gráfica estadística de paciente.
- Visualización de mensajes informativos entre niveles de la terapia cognitiva.
- Reinicio de avances del paciente.
- Cambio de usuario en la aplicación.

5.3 Casos de uso

Es usual, a la hora de definir un nuevo sistema software, recurrir a un análisis y diseño orientado a objetos a partir de los requisitos de usuario, denominado "casos de uso" [85]. Dentro de todas las posibilidades que ofrece este sistema, se presentará la parte más visual e intuitiva para el lector mediante un diagrama UML en el que se definen dos elementos fundamentales:

- **Actores**. Representan un rol de usuario que interactúa con el sistema.
- **Casos de uso**. Son las tareas concretas que se realizan tras la orden de un agente externo, ya sea un actor u otro caso de uso.

La Figura 62 muestra el diagrama de casos de uso que han sido representados mediante óvalos con una descripción textual en su interior. Por su parte, el sistema HealthCoach se representa por un

cuadro que alberga todos los casos de uso. Fuera del sistema aparecen los dos actores que interactúan con el sistema, paciente y responsable.

La funcionalidad que representa cada caso de uso es descrita a continuación:

- **Registro de usuario.** Registra al usuario en el sistema, procedimiento compartido por pacientes y responsables.
- **Seleccionar usuario.** Identifica al usuario entre un listado de todos los asociados al terminal desde el que se accede. Aplica a pacientes y responsables.
- **Cambiar de usuario.** Permite volver a realizar el proceso de identificación de usuario. Aplica a pacientes y responsables.
- **Seleccionar paciente.** Selecciona al paciente de entre los asociados al responsable identificado. Aplica únicamente a responsables.
- **Consultar estadísticas de paciente.** Consulta las estadísticas generadas con el desarrollo de la terapia. Aplica solo a responsables.
- **Iniciar/continuar terapia cognitiva.** Permite comenzar la terapia en el nivel inicial o continuar con ella en el punto que se dejó. Aplica a pacientes.
- **Contestar tarjetas mediante el habla.** Proporciona respuesta a las tarjetas presentadas por medio de la voz del paciente. Aplica solo a pacientes.
- **Guardar avances y salir.** Guarda en el sistema el progreso y estado del usuario para proceder a salir de la aplicación. Aplica únicamente a pacientes.
- **Mostrar grado enfermedad paciente.** Aplica solo en responsables.
- **Restablecer avances de paciente.** Elimina los datos asociados a la terapia del paciente seleccionado. Aplica a responsables.
- **Cambiar de paciente.** Permite volver a realizar el proceso de selección de paciente de entre los asignados al responsable.

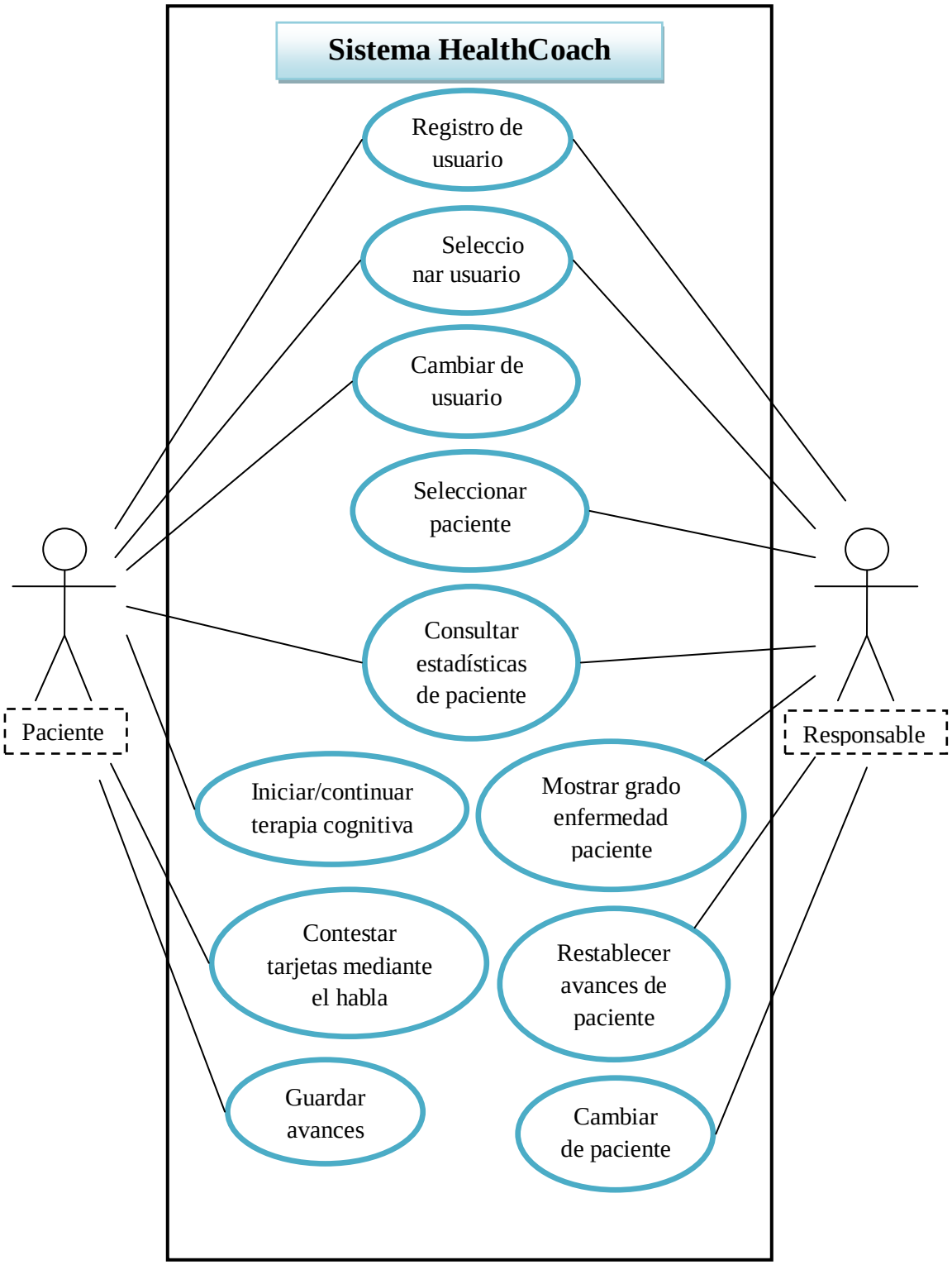


Figura 62. Casos de uso para paciente y responsable en el sistema HealthCoach

En cuanto a los casos de uso del servidor del sistema, al que se conectan las aplicaciones de usuario, se ha elaborado el diagrama de la Figura 63 para representarlos.

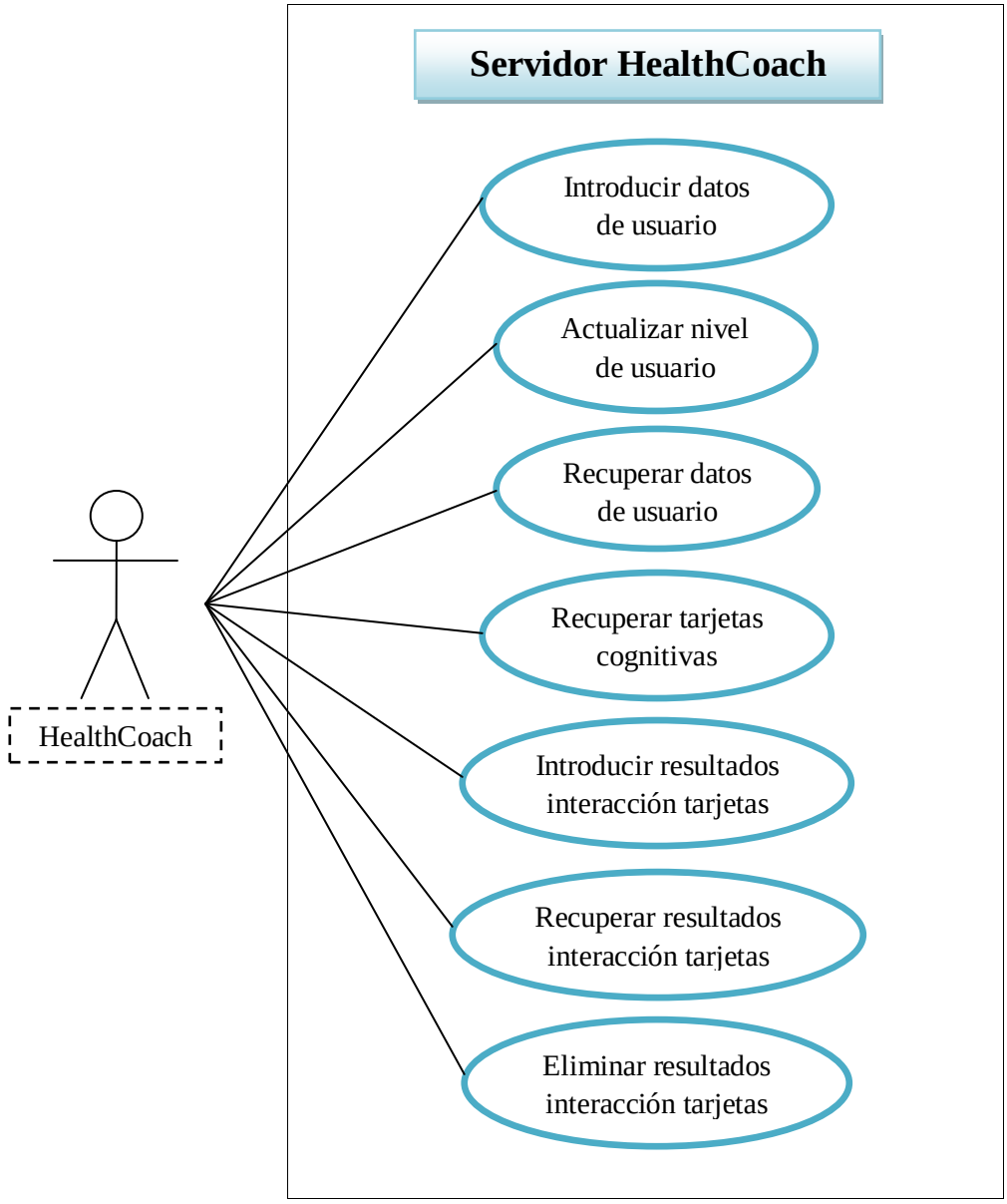


Figura 63. Casos de uso del servidor HealthCoach

En esta ocasión, existe un único actor que interactúa con los casos de uso proporcionados por el servidor, la propia aplicación *HealthCoach* en sus dos roles de usuario. Los casos de uso mostrados son:

- **Introducir datos de usuario.** Guarda la información del nuevo usuario registrado.
- **Actualizar nivel de usuario.** Actualiza el nivel de tarjetas alcanzado por el usuario en el progreso de la terapia cognitiva.
- **Recuperar datos de usuario**. Recupera cualquiera de los datos asociados al perfil del usuario.
- **Recuperar tarjetas cognitivas.** Accede a la información de cada una de las tarjetas cognitivas disponibles para ser presentadas al usuario en la terapia.

- **Introducir resultados de interacción con tarjetas.** Introduce información relativa a las respuestas del usuario a cada tarjeta cognitiva.
- **Recuperar resultados de interacción con tarjetas.** Recupera datos relativos a las respuestas generadas en la interacción con la terapia cognitiva.
- **Eliminar resultados de interacción con tarjetas.** Suprime datos referentes a la interacción del usuario con las tarjetas cognitivas.

5.4 Arquitectura general

El sistema que conforma la totalidad de **HealthCoach** está basado en una arquitectura cliente-servidor, en el que los dispositivos móviles que poseen las dos aplicaciones Android desarrolladas, una de pacientes y la otra de responsables, constituyen los clientes que consumirán los servicios proporcionados en el servidor remoto. En la Figura 64 puede observarse una representación gráfica de la arquitectura expuesta.

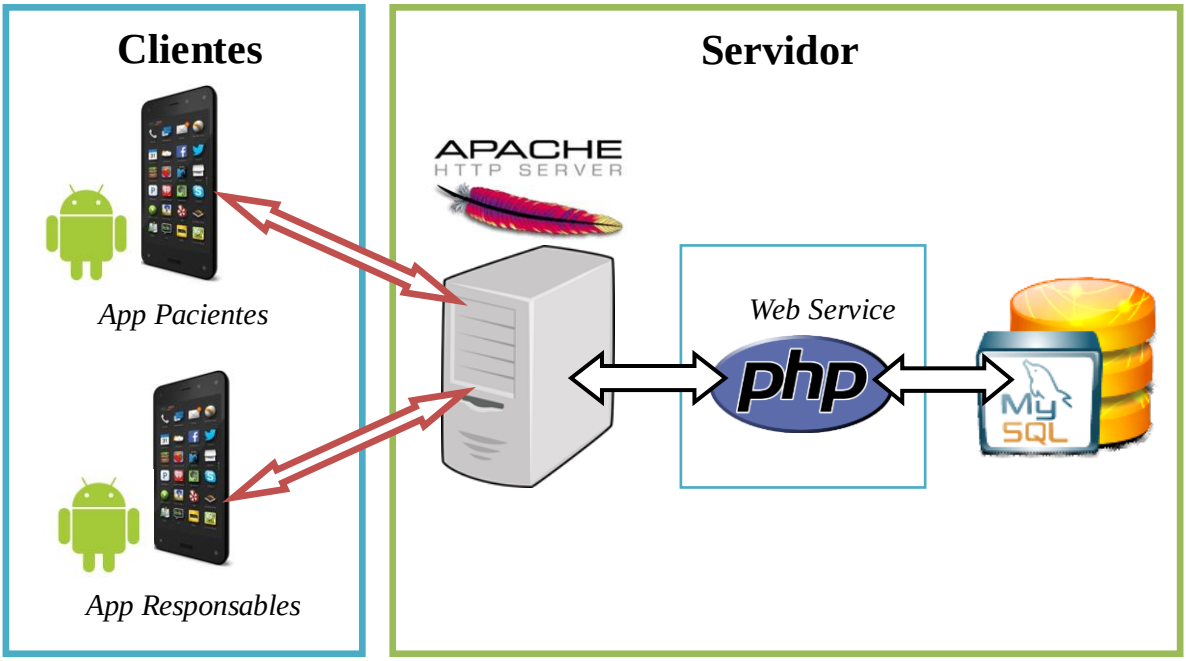


Figura 64. Arquitectura general del sistema HealthCoach

5.4.1 Aplicaciones cliente

Para lograr sus objetivos, *HealthCoach* se divide en dos aplicaciones móviles Android bien diferenciadas:

HealthCoach

La primera, destinada a los ***pacientes*** y denominada ***HealthCoach***, propone la realización de ejercicios terapéuticos de diversos ámbitos cognitivos mediante tarjetas multimodales que combinan interacción visual, táctil y de lenguaje natural gracias al reconocimiento automático del habla y síntesis

de texto a voz. Además, se complementa con un proceso de **adaptabilidad** que permite establecer los ejercicios más adecuados para el enfermo, dejando de presentarle algunos o potenciando la realización de otros, así como un módulo estadístico que analiza la evolución de la terapia y envía sus resultados por correo electrónico a los responsables del paciente.

HealthCoach Admin

La segunda aplicación, llamada ***HealthCoach Admin***, cumple una faceta ***administrativa*** para el médico, terapeuta o familiar a cargo del paciente, es decir, permite acceder a la información detallada que genera el paciente al realizar las tarjetas de terapia cognitiva, como son las estadísticas de uso o el grado estimado de la enfermedad, así como reiniciar la terapia.

Ambas aplicaciones necesitan una **conexión a Internet** desde el dispositivo móvil, ya sea a través de Wi-Fi o datos móviles, para comunicarse con el servidor remoto y poder recuperar o grabar información involucrada en el funcionamiento del sistema. Este intercambio de datos se realiza utilizando el protocolo **HTTP** y el estándar **JSON**, como se detalla en el **Capítulo 3** de esta memoria.

Estas aplicaciones cliente se basan en los componentes descritos en el **Capítulo 4**. En el manual de usuario, contenido en el **Anexo A**, puede observarse la interfaz de usuario a través del flujo de pantallas de cada aplicación.

5.4.2 Servidor

El servidor remoto al que se conectan los clientes del sistema propuesto, es decir, los dispositivos móviles que utilizan la aplicación de pacientes y/o responsables, realiza la función de proveer los datos necesarios en los procesos de identificación, registro, carga de tarjetas, estadísticas de usuario, etc. Pero también realiza la importante función de persistir todas las respuestas proporcionadas por el paciente en el desarrollo de la terapia de usuario, para así poder llevar a cabo la adaptabilidad de la dificultad y tipología de las tarjetas de ejercicios propuestas.

Para lograr su propósito, la parte servidor se compone de tres componentes:

- **Servidor de aplicaciones Apache** que escucha las peticiones HTTP entrantes y alberga el módulo PHP descrito a continuación.
- **Módulo PHP** para albergar la API REST que gestiona la comunicación entre las aplicaciones cliente y la base de datos remota.
- **Base de datos MySQL** para gestionar la persistencia de la información involucrada en el sistema.

Estos componentes se instalarán en un ordenador de bajo coste y consumo energético denominado **Raspberry Pi**, que realizará la función de servidor 24/7. Este servidor es gestionado por el sistema operativo **Raspbian**, basado en la famosa distribución Debian de Linux y optimizado para el hardware existente en Raspberry Pi. La Figura 65 muestra la imagen del dispositivo Raspberry Pi junto con los componentes software instalados en él.

Figura 65. Hardware y software utilizados en la parte servidor del sistema

El **Anexo A** describe en detalle los procesos de instalación y configuración del sistema operativo Raspbian, servidor de aplicaciones Apache, módulo PHP y gestor de bases de datos MySQL realizados en el dispositivo Raspberry Pi.

5.4.2.1 API REST

El servidor del sistema HealthCoach realiza la importante función de albergar el servicio o API REST, desarrollado en lenguaje PHP utilizando el framework Slim para facilitar el desarrollo del mismo.

Para implementar el servicio web, se utilizará un fichero index.php que contenga la lógica para interceptar las peticiones REST y generar el fichero JSON con los datos solicitados o realizar la inserción/modificación deseada en la base de datos MySQL.

En el **Capítulo 6** de esta memoria, referente a la implementación del sistema se profundiza en el desarrollo y procedimientos de interacción con esta API.

5.4.2.2 Persistencia

La implementación de la persistencia de datos en el sistema **HealthCoach** recae en un gestor de base de datos **MySql 5**, que provee licencia libre y gratuita para usos no lucrativos como es el caso del presente proyecto. En él se ha creado una base de datos llamada ***proyectofincarrera*** que almacena la información necesaria referente a los datos personales de los usuarios, las tarjetas a mostrar en la

terapia cognitiva y todos los registros generados por el usuario en el proceso de interacción con las tarjetas presentadas.

La estructura de *proyectofincarrera* se compone de tres tablas como puede observarse en el esquema relacional de la Figura 66. La tabla ***USERS*** recopila toda la información asociada al usuario, posee una serie de datos proporcionados directamente por el usuario en el momento del registro en la aplicación, como son nombre, apellidos, email, pero también otros recopilados o actualizados de forma automática por el sistema, como son el campo **IMEI** (código univoco del terminal móvil) y **DEGREE** (representa el grado de enfermedad del paciente).

La tabla ***CARDS*** modela las tarjetas que serán utilizadas en la terapia cognitiva del paciente mediante campos que clasifican la tipología, recogen los enunciados y respuestas correctas, URL de la imagen a mostrar, etc., además de contener el identificador de clave primaria IDC para mantener la integridad. En último lugar, la tabla ***RECORDS*** recopila los registros generados desde la aplicación de pacientes cuando el usuario contesta exitosa o erróneamente cada una de las tarjetas presentadas, es decir, supone un log de respuestas.

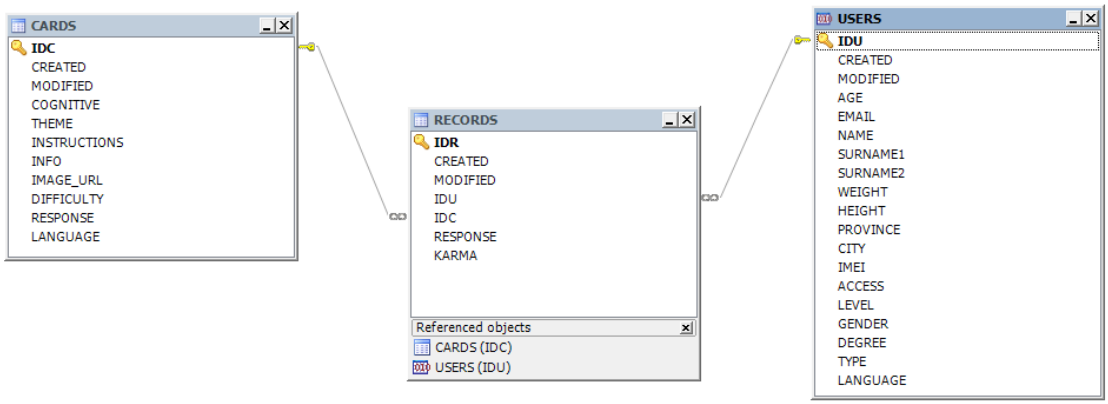


Figura 66. Diagrama relacional de BBDD obtenido con Toad for MySQL

Cada tabla dispone de una clave primaria que actúa como identificador de los elementos registrados en ellas, siendo *IDC* el identificador de tarjeta utilizado en la tabla *CARDS*, *IDR* el identificador de registro de la tabla *RECORDS* e *IDU* el identificador de usuario dentro de la tabla *USERS*.

A continuación se muestra el script SQL utilizado para crear la base de datos y las tres tablas que conforman la persistencia del sistema HealthCoach. Para generarlo, se ha recurrido a la herramienta *Toad for MySQL* descrita en el **Apartado 3.2.4** de esta memoria.

```
-- Volcando estructura de base de datos para proyectofincarrera
CREATE DATABASE IF NOT EXISTS `proyectofincarrera` /*!40100 DEFAULT CHARACTER
SET utf8 COLLATE utf8_unicode_ci */;
USE `proyectofincarrera`;
-- Volcando estructura para tabla proyectofincarrera.CARDS
CREATE TABLE IF NOT EXISTS `CARDS` (
  `IDC` int(11) NOT NULL AUTO_INCREMENT,
  `CREATED` datetime DEFAULT NULL,
  `MODIFIED` datetime DEFAULT NULL,
  `COGNITIVE` varchar(15) COLLATE utf8_unicode_ci NOT NULL,
  `THEME` varchar(15) COLLATE utf8_unicode_ci NOT NULL,
  `INSTRUCTIONS` varchar(250) COLLATE utf8_unicode_ci NOT NULL,
  `INFO` varchar(250) COLLATE utf8_unicode_ci NOT NULL,
  `IMAGE_URL` varchar(250) COLLATE utf8_unicode_ci DEFAULT NULL,
  `DIFFICULTY` int(1) NOT NULL,
  `RESPONSE` varchar(250) COLLATE utf8_unicode_ci DEFAULT NULL,
  `LANGUAGE` varchar(25) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`IDC`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
-- Volcando estructura para tabla proyectofincarrera.RECORDS
CREATE TABLE IF NOT EXISTS `RECORDS` (
  `IDR` int(11) NOT NULL AUTO_INCREMENT,
  `CREATED` datetime DEFAULT NULL,
  `MODIFIED` datetime DEFAULT NULL,
  `IDU` int(11) NOT NULL,
  `IDC` int(11) NOT NULL,
  `RESPONSE` varchar(250) DEFAULT NULL,
  `KARMA` varchar(1) DEFAULT NULL,
  PRIMARY KEY (`IDR`),
  KEY `FK_IDUSER_RECORD` (`IDU`),
  KEY `FK_IDCARD_RECORD` (`IDC`),
  CONSTRAINT `FK_IDCARD_RECORD` FOREIGN KEY (`IDC`) REFERENCES `CARDS` (`IDC`),
  CONSTRAINT `FK_IDUSER_RECORD` FOREIGN KEY (`IDU`) REFERENCES `USERS` (`IDU`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- Volcando estructura para tabla proyectofincarrera.USERS
CREATE TABLE IF NOT EXISTS `USERS` (
  `IDU` int(11) NOT NULL AUTO_INCREMENT,
  `CREATED` datetime DEFAULT NULL,
  `MODIFIED` datetime DEFAULT NULL,
  `AGE` int(3) DEFAULT NULL,
  `EMAIL` varchar(100) NOT NULL,
  `NAME` varchar(250) NOT NULL,
  `SURNAME1` varchar(250) NOT NULL,
  `SURNAME2` varchar(250) NOT NULL,
  `WEIGHT` int(3) DEFAULT NULL,
  `HEIGHT` int(3) DEFAULT NULL,
  `PROVINCE` varchar(15) NOT NULL,
  `CITY` varchar(15) DEFAULT NULL,
  `IMEI` varchar(15) NOT NULL,
  `ACCESS` datetime DEFAULT NULL,
  `LEVEL` int(1) NOT NULL,
  `GENDER` char(1) NOT NULL,
  `DEGREE` int(1) DEFAULT NULL,
  `TYPE` varchar(1) NOT NULL,
  `LANGUAGE` varchar(25) NOT NULL,
  PRIMARY KEY (`IDU`)
   ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Figura 67. Script SQL que genera la base de datos y tablas necesarias en el sistema

5.5 Diagrama de clases

A continuación se presentan los diagramas con las clases Java involucradas en las aplicaciones HealhCoach (Figura 68) y HealthCoach Admin (Figura 69). Muestran una versión simplificada por razones de espacio, en la que cada caja representa una clase identificada por el nombre que alberga en su interior. En el capítulo de implementación se explican en mayor profundidad los principales métodos que implementan así como la relación existente entre las clases.

Aunque se explicarán en mayor detalle en el capítulo referente a la implementación de los módulos del sistema, se procede a listar las principales clases junto con una breve descripción.

- **AndroidSplashActivity.java:** es la clase que se muestra al arrancar la aplicación, mostrando el logo de HealthCoach a pantalla completa durante un segundo.

- **BarPlotActivity.java:** es la actividad encargada de generar la gráfica estadística en formato de barras.

- **BarPlotSerieActivity.java:** actividad que genera la gráfica estadística basada en series.

- **Card.java:** esta clase modela los objetos tarjeta, definiendo sus atributos y métodos de acceso.

- **CardError.java:** es una actividad que gestiona la pantalla mostrada cuando se produce un "error" al recuperar las tarjetas disponibles para el usuario, cuya razón es que ya no existen más tarjetas en base de datos a realizar.

- **Degree.java:** esta actividad calcula y muestra en pantalla el grado estimado de la enfermedad de Alzheimer del paciente.

- **DepthPageTransformer.java:**

- **HiScreen.java:** Actividad que muestra la pantalla de bienvenida personalizada con los datos del usuario y gestiona el menú de opciones para realizar la navegación a otras actividades.

- **Level.java:** es la actividad que muestra la pantalla informativa entre niveles de la terapia cognitiva, para informar de la consecución de un nivel y permitir avanzar al siguiente, volver a la pantalla de bienvenida o visualizar las estadísticas.

- **MyAdapter.java:** es la clase que ejerce de adaptador de las listas de usuarios mostradas en las pantallas de identificación, definiendo la estructura de la información de paciente mostrada.

- **MyUserAdminAdapter.java:** es el adaptador encargado de establecer el formato de la lista que muestra a los responsables que van a identificarse en un dispositivo.

- **Record.java:** representa los registros con las respuestas asociadas a una tarjeta y un usuario, así como si posee un karma positivo o negativo, es decir, éxito o fallo en la respuesta.

- **Register.java:** es la clase que gestiona el registro del usuario en el sistema.

- **ScreenSlidePageFragment.java:** esta importante clase extiende de la clase Fragment y contiene la estructura de cada una de las tarjetas generadas para la terapia cognitiva. Posee un proceso de generación automática de la interfaz en función del tipo de tarjeta. Además contiene la lógica para llevar a cabo el proceso de reconocimiento automático del habla del paciente para decidir si una respuesta ha sido correcta o errónea.

- **ScreedSlidePagerActivity.java**: es la actividad más importante de la aplicación. Se encarga de obtener la información de las tarjetas cognitivas y aplicar el proceso de adaptabilidad para definir las que finalmente se mostrarán al paciente. Después crea dichas tarjetas en objetos *Fragment* para mostrarlas de forma paginada. Es la responsable de realizar el registro en base de datos de las respuestas generadas en la terapia cognitiva. Por último, gestiona el envío de correos electrónicos a los responsables informando sobre eventos en la terapia de su paciente.

- **ServiceHandler.java:** esta clase realiza las llamadas HTTP a las url definidas en el servicio web REST del servidor remoto y retorna la respuesta obtenida.

- **SimplePieChartActivity.java:** esta actividad se encarga de generar la gráfica estadística en forma de tarta.

- **StatisticsLauncher.java:** define la actividad que muestra y gestiona el uso de la lista de gráficas estadísticas disponibles para el usuario.

- **User.java:** clase que representa a un usuario, con todos los datos introducidos en el registro y algunos adicionales utilizados por el sistema.

- **UserSelector.java:** es la actividad que gestiona la selección de usuario entre los disponibles en el dispositivo para identificarse en el sistema.

- **UserSelectorPatient.java:** es la clase que recupera y muestra la lista de pacientes asociados a un responsable identificado en el sistema.

- **ZoomOutPageTransformer.java:** es la clase que establece un efecto en la transición entre tarjetas.

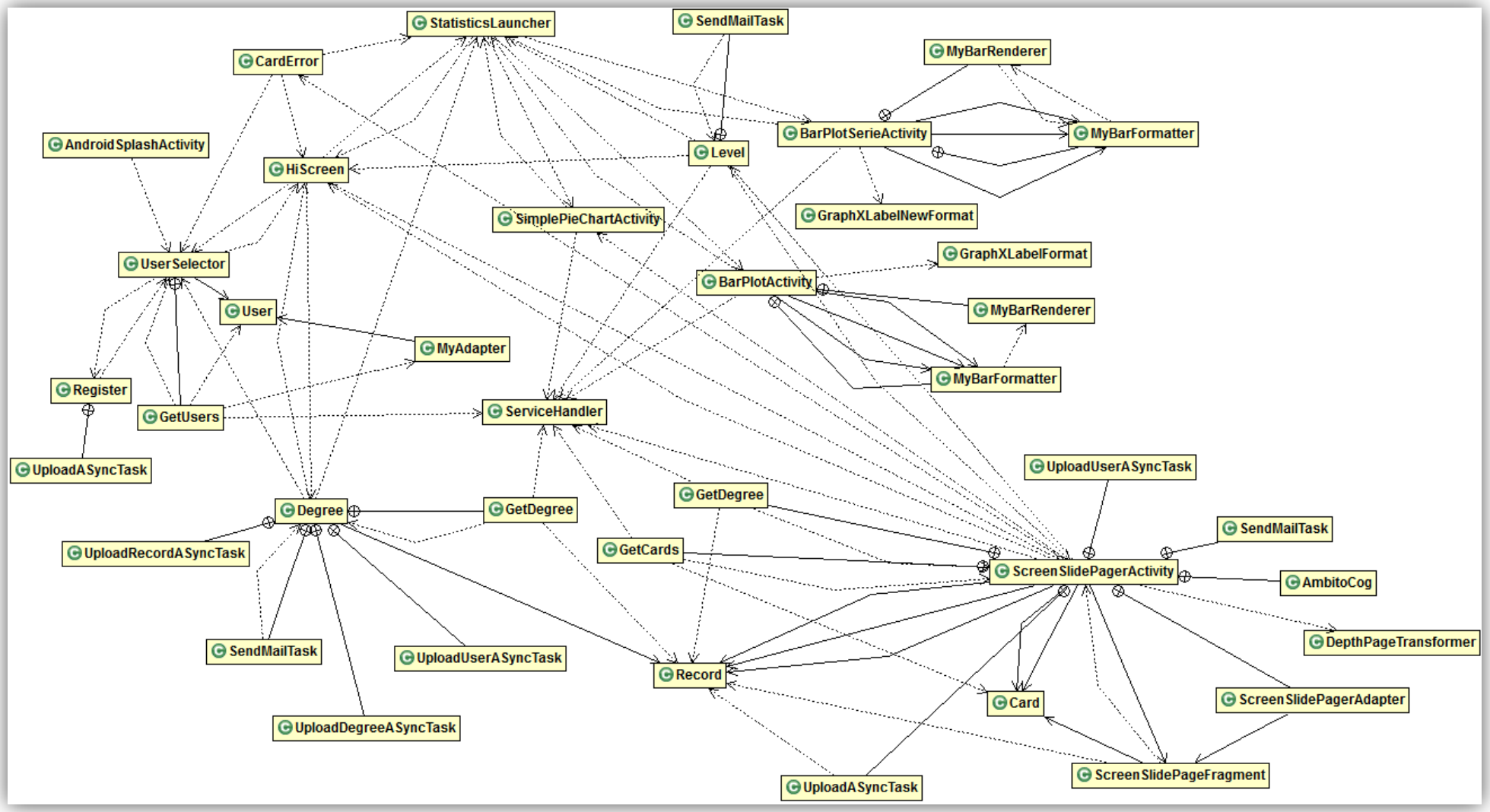


Figura 68. Diagrama de clases de aplicación HealthCoach

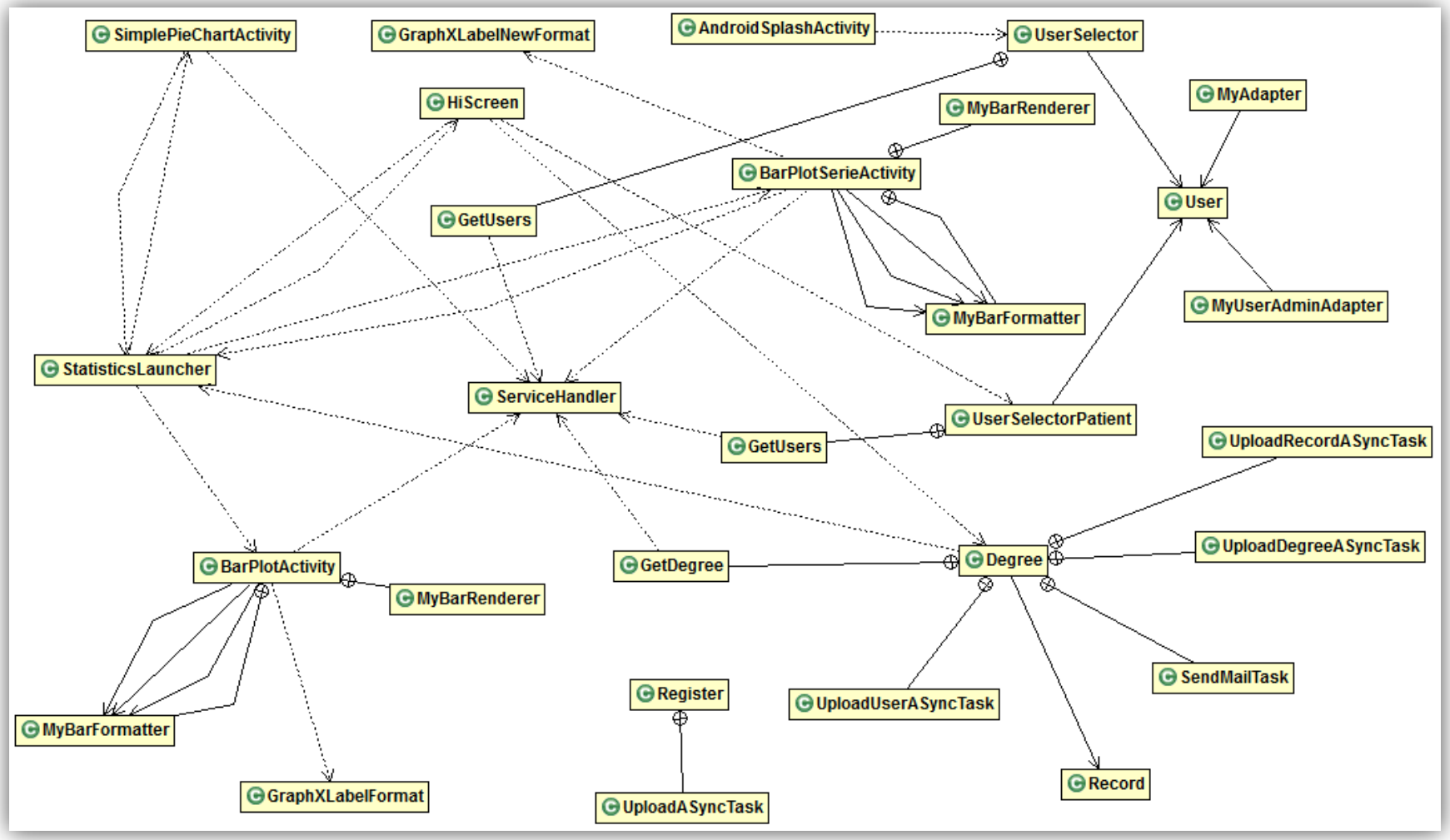


Figura 69. Diagrama de clases de HealthCoach Admin

5.6 Diseño de la interfaz de usuario

La interfaz de usuario es el componente que facilita la interacción de los usuarios con la aplicación, por tanto debe adaptarse a las características y requerimientos de esos usuarios. Basándose en el propósito del sistema **HealthCoach**, destinado a pacientes de Alzheimer generalmente de edad avanzada y con una alta probabilidad de no haber tenido contacto previo con tecnologías digitales, la interfaz debe ser intuitiva en su manejo y fácilmente legible. Estas son las premisas que han establecido la filosofía de diseño simple y práctico de la interfaz de usuario.

El diseño de la interfaz de usuario se ha realizado con la ayuda de la herramienta ***Fluid UI*** [86] un editor online con plantillas y componentes que permiten simular una pantalla de dispositivo Android. Supone un primer esbozo de la apariencia final de la aplicación así como de las funcionalidades, debido a ello puede estar sujeto a cambios en la fase de implementación. Las modificaciones suelen deberse a problemáticas en la fase de implementación o incluso imposibilidad de abordar elementos del diseño o funcionalidades por la plataforma o *frameworks* utilizados. La interfaz y características definitivas pueden consultarse en el manual de usuario del **Anexo B**. A continuación, se expone el diseño inicial de las pantallas de las aplicaciones Android.

5.6.1 Pantalla de registro

La pantalla de registro, mostrada en la Figura 70, debe permitir al usuario introducir los datos de perfil necesarios que se guardarán en el sistema. Como particularidad, esta pantalla se utilizará tanto en la aplicación HealthCoach como HealthCoach Admin, en la primera de ellas el correo electrónico introducido pertenecerá al responsable del usuario y en la segunda el propio correo del responsable a registrar. En HealthCoach Admin tampoco será necesario el desplegable con el idioma en el que se presentarán las tarjetas.

Figura 70. Diseño preliminar pantalla de registro de usuario

5.6.2 Pantalla selección usuario

La pantalla de acceso a las aplicaciones de pacientes y responsables es un listado de los usuarios registrados anteriormente desde ese dispositivo móvil, identificados gracias a la asociación automática a través del código IMEI del terminal en el momento del registro. Cada elemento de la lista de usuarios muestra el nombre y nivel actual en la terapia cognitiva. Desde la parte inferior de la aplicación existirá un enlace al registro de nuevo usuario, al que se accederá automáticamente si no se recupera ningún usuario. La Figura 71 muestra el diseño de esta pantalla.

Figura 71. Diseño de pantalla de selección de usuario

5.6.3 Pantalla selección de paciente

Los responsables necesitan una pantalla que liste todos los pacientes que tienen asociados. Muestra en cada elemento listado el nombre del paciente y el nivel de dificultad alcanzado en la terapia de tarjetas cognitivas. Puede observarse una ilustración del diseño en la Figura 72.

Figura 72. Diseño de pantalla de selección de paciente

5.6.4 Pantalla de bienvenida

Una vez el usuario se identifica en el sistema, se le presenta una pantalla de bienvenida que cumple las funciones de menú principal, el cuál le permite comenzar o continuar la terapia cognitiva, acceder al menú de gráficas estadísticas, cambiar de usuario en el dispositivo o salir de la aplicación. Esta pantalla se utiliza en las dos aplicaciones, **HealthCoach** y **HealthCoach Admin**.

Figura 73. Diseño de pantalla de bienvenida y menú principal

5.6.5 Pantalla tarjeta ejercicios

Para representar las pantallas de la aplicación **HealthCoach** que muestran las tarjetas cognitivas, se han diseñado tres plantillas que abarcan los tipos de estructura necesarios. Como denominador común, todas presentan en su parte inferior un botón con la imagen de un micrófono, cuya pulsación inicia el reconocimiento automático del habla, y una vez finalizado el proceso se muestra la respuesta en pantalla.

En el **Apartado 5.7**, se profundiza en el diseño de los diversos tipos de ámbito cognitivos que albergan las tarjetas.

5.6.5.1 Con imágenes

Independientemente del ámbito cognitivo que representen, existen tarjetas que muestran una imagen en pantalla, siempre precedida de una pregunta o enunciado referidos a ella. Las tarjetas de la Figura 74 son un ejemplo de este tipo de pantalla.

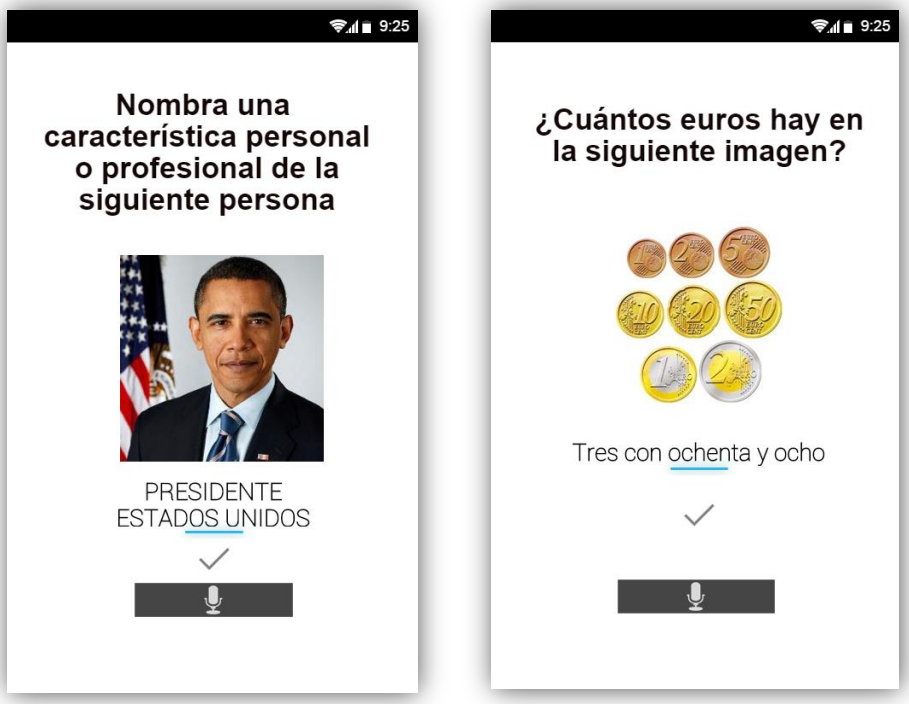


Figura 74. Diseños de tarjeta cognitiva con imagen

5.6.5.2 Enunciado simple

Estas tarjetas muestran únicamente un enunciado o pregunta a la que debe contestar el paciente, denominado enunciado principal. Es el caso más sencillo, con un diseño como el mostrado en la Figura 75.

Figura 75. Diseños de tarjeta cognitiva con descripción simple

5.6.5.3 Doble enunciado

Existen pantallas con un doble enunciado. Complementando al enunciado principal, en el que se indica la acción a realizar, existe un texto adicional para definir el ejercicio, como muestran los diseños de la Figura 76.

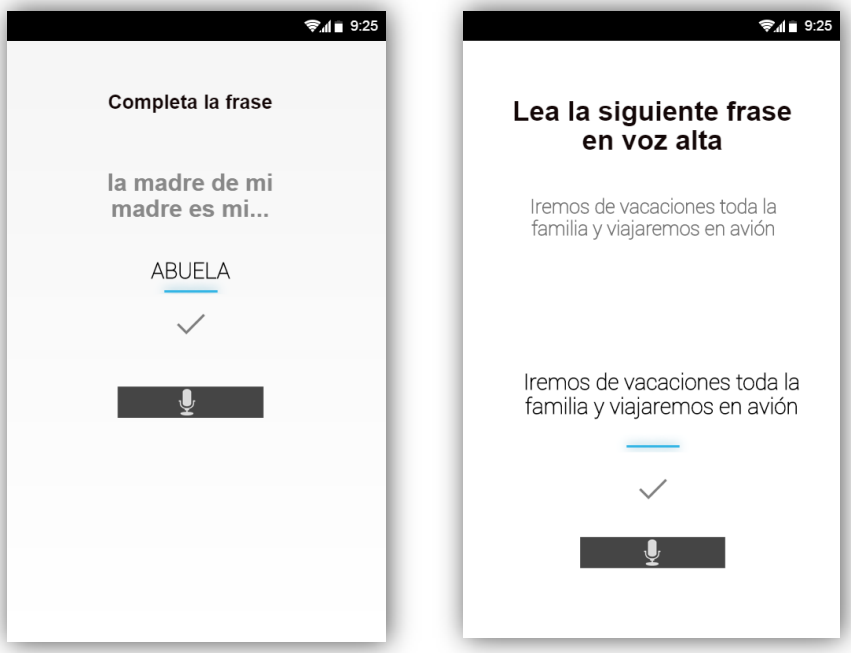


Figura 76. Diseño de tarjeta con doble descripción

5.6.6 Pantalla informativa entre niveles

Al superar todas las tarjetas pertenecientes a un mismo nivel de la terapia cognitiva, seleccionadas gracias al algoritmo de adaptabilidad de usuario, debe mostrarse una pantalla informativa sobre la finalización de ese nivel. Adicionalmente, deberá permitir al usuario continuar con las tarjetas del siguiente nivel, salir de la terapia o visualizar sus estadísticas hasta ese momento. El diseño de esta pantalla puede observarse en la Figura 77.

Figura 77. Diseño de pantalla informativa en terapia cognitiva

5.6.7 Pantalla de listado de estadísticas

Debido a la posibilidad de presentar al usuario un conjunto de gráficas con estadísticas, surge la necesidad de diseñar una pantalla que permita la visualización de todas ellas en forma de listado, así como la posibilidad de volver a la pantalla de bienvenida (**Apartado 5.6.4**). Este listado será accesible desde las aplicaciones *HealthCoach* y *HealthCoach Admin*. La Figura 78 muestra el diseño preliminar de esta pantalla.

Figura 78. Diseño de pantalla para menú de estadísticas

5.6.8 Pantalla cálculo estadio enfermedad

Los responsables pueden consultar el grado o estadio estimado de Alzheimer de cada uno de sus pacientes. El diseño de la pantalla que lo muestra se refleja en la Figura 79. Como puede observarse, además de presentar un mensaje indicando de forma numérica el estadio, presenta las opciones de Continuar para volver al menú principal de la aplicación *HealthCoach Admin* y resetear los datos asociados al avance de la terapia del paciente que se está visualizando.

Figura 79. Diseño de pantalla de estadio estimado de Alzheimer

5.7 Tarjetas cognitivas

El propósito final de este proyecto es el de diseñar un sistema que se adapte al tratamiento de diferentes dolencias, en el caso práctico de esta memoria, que realice una personalización de los ejercicios terapéuticos disponibles para llegar a diagnosticar el grado/estadio de la enfermedad de Alzheimer en la que se encuentra el usuario. Por ello, se han diseñado una serie de **tarjetas** para presentar al usuario a través de la interfaz de *HealthCoach*, siendo necesario realizar una serie de inputs o acciones que serán indicadas por pantalla, registrando los resultados de acierto y error para una progresiva adaptación del tipo de ejercicios a realizar para favorecer en una correcta terapia.

Los ejercicios que contendrán las **tarjetas** se clasifican por:

- Función cognitiva a rehabilitar.
- Nivel de dificultad.

Por otro lado, la realización de los ejercicios mediante fichas de estimulación cognitiva tiene como objetivos:

- Mejorar las habilidades que en el proceso cognitivo se deterioran.
- Mantener las habilidades intelectuales, conservándolas al máximo tiempo posible, con la finalidad de preservar su autonomía.
- Hacer pasar un rato agradable al paciente.
- Estimular los procesos cognitivos: memoria, atención, etc.
- Mejorar su calidad de vida.

De cara a potenciar las funciones cognitivas de los pacientes con Alzheimer y basándose en el estudio sobre las técnicas terapéuticas llevado a cabo en el **Apartado 2.6.3**, se presenta una muestra del diseño de las tarjetas de ejercicios clasificadas según la función cognitiva que pretenden estimular.

5.7.1 Memoria

Las tarjetas que representan ejercicios terapéuticos para la estimulación de la memoria, se diseñan en función del tipo específico que pretenden estimular, ya sea inmediata, reciente o episódica.

5.7.1.1 Memoria inmediata y reciente

Los recuerdos inmediatos y recientes se ejercitan mediante tarjetas de doble enunciado (**Apartado 5.6.5.3**) o enunciado simple (**Apartado 5.6.5.2**). En la Figura 80 se muestra un ejemplo del primer tipo, donde después del enunciado con las indicaciones para realizar el ejercicio se presenta una historia que debe leer el paciente o terapeuta para posteriormente contestar a la pregunta formulada en referencia a la historia.

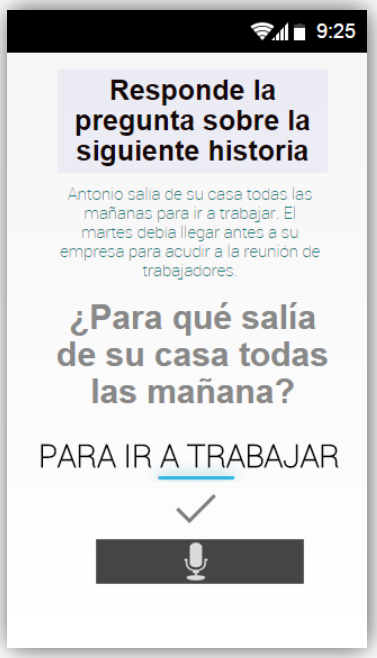


Figura 80. Memoria episódica en tarjeta cognitiva

5.7.1.2 Memoria remota

Este tipo de memoria engloba la memoria biográfica personal (episódica), así como los acontecimientos y conocimientos adquiridos (semántica). Para representar estos ejercicios se recurre al tipo de tarjeta de enunciado simple (**Apartado 5.6.5.2**), ya que únicamente es necesario un enunciado adicional a las instrucciones para realizarlo, como puede observarse en la Figura 81.

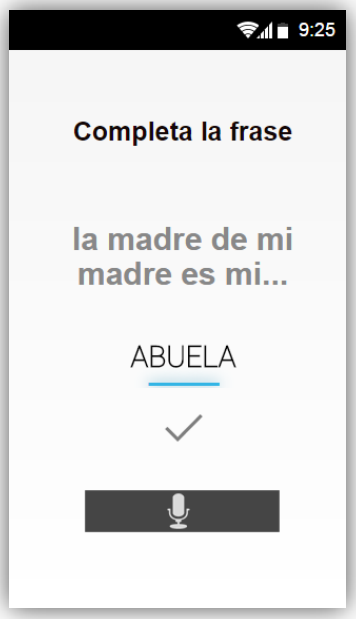


Figura 81. Tarjeta cognitiva destinada a la memoria episódica a largo plazo

5.7.2 Orientación

La capacidad de orientación espacio-temporal y de la persona se trabaja mediante tarjetas cognitivas con imágenes (**Apartado 5.6.5.1**) y de enunciado simple (**Apartado 5.6.5.2**).

5.7.2.1 Tiempo

La orientación temporal se estimula mediante tarjetas que se apoyan en elementos gráficos, como relojes o calendarios, para facilitar al paciente la realización del ejercicio. La Figura 82, muestra un ejemplo de ejercicio de orientación temporal a través de un reloj.

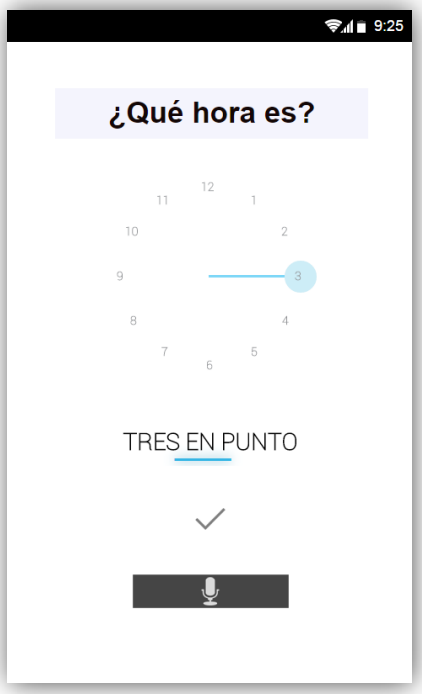


Figura 82. Diseño de tarjeta para la orientación temporal

5.7.2.2 Espacio

Mediante las tarjetas destinadas a favorecer la orientación espacial se pretende que el paciente tome conciencia de su situación en el espacio. Las tarjetas de enunciado simple, como la mostrada en la Figura 83, son adecuadas para la presentación del ejercicio.

Figura 83. Diseño de tarjeta de orientación espacial

5.7.2.3 Persona

Los ejercicios de orientación personal favorecen la activación de la memoria autobiográfica. Ejercicios como el mostrado en la Figura 84, representan el diseño elaborado mediante tarjetas de enunciado simple (**Apartado 5.6.5.2**).

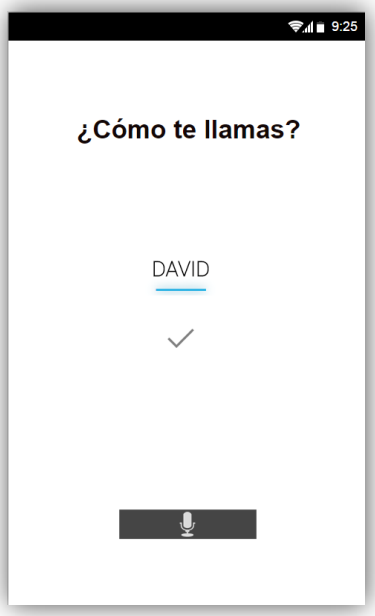


Figura 84. Diseño de tarjeta de orientación de persona

5.7.3 Gnosias

Mediante el diseño de las tarjetas de ejercicios que se centran en las gnosias, se pretende mejorar la percepción de las formas y las características físicas de objetos y personas.

5.7.3.1 Reconocimiento de objetos

El diseño de tarjeta realizado para medir y trabajar el reconocimiento de objetos se basa en las tarjetas con imágenes presentadas en el **Apartado 5.6.5.1**. Un ejercicio tipo es el mostrado en la Figura 85, donde el paciente debe pronunciar el nombre del objeto mostrado en la imagen presentada.

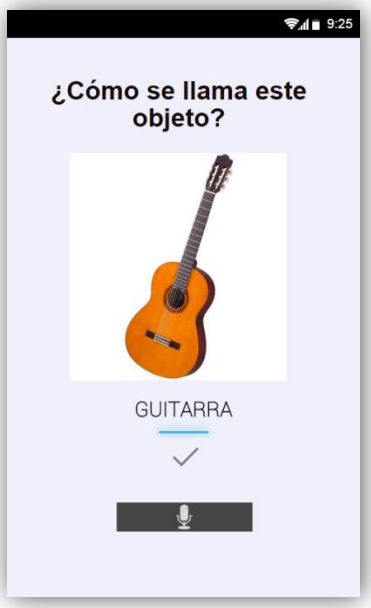


Figura 85. Diseño de tarjeta con reconocimiento de objetos

5.7.3.2 Reconocimiento de caras

La terapia sobre las capacidades de reconocimiento facial se lleva a cabo mediante tarjetas con imágenes (**Apartado 5.6.5.1**).

Hombres y mujeres famosos

Para el reconocimiento de personajes famosos se utilizan fotografías donde se les pueda identificar de forma unívoca, como es el caso de la Figura 86.

Figura 86. Reconocimiento de caras en tarjeta cognitiva

Característica personal o profesional

Otro ejercicio dentro de la terapia de gnosias consiste en preguntar al paciente por alguna característica personal o profesional de la persona mostrada en la imagen, como se observa en la Figura 87.

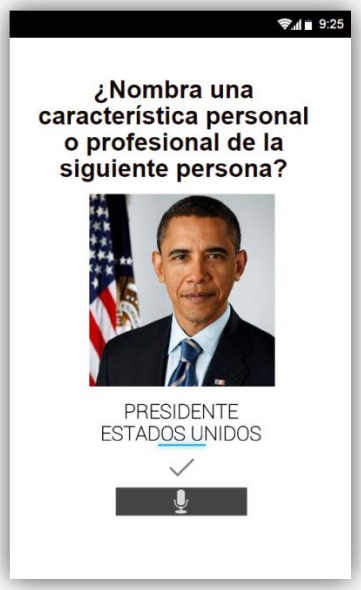


Figura 87. Características personales en tarjeta cognitiva

Estados de ánimo

Las tarjetas cognitivas con imágenes permiten preguntar al paciente a cerca del estado de ánimo en el que se encuentra el individuo mostrado en la imagen. El ejemplo de la Figura 88, muestra la cara triste de un bebé.

Figura 88. Diseño de tarjeta para reconocimiento de estados de ánimo

5.7.3.3 Representación mental del espacio y objetos

Este tipo de ejercicio cognitivo es utilizado para fomentar la representación mental de lugares. Por lo tanto, es necesario el uso de imágenes como la utilizada en el ejemplo de diseño de la Figura 89, donde se muestra una fotografía de un parque lleno de gente.

Figura 89. Representación espacial en tarjeta cognitiva

5.7.3.4 Formas

El reconocimiento de formas requiere un diseño de tarjeta que permita la visualización de imágenes, como la mostrada en la Figura 90, por lo tanto se basa en el tipo de tarjetas analizado en el **Apartado 5.6.5.1**.

Figura 90. Tarjeta de reconocimiento de formas

5.7.3.5 Color

El diseño de las tarjetas utilizadas en el reconocimiento de colores persigue mejorar la percepción cromática del paciente. La Figura 91 muestra un ejemplo de diseño, basado en una tarjeta de enunciado simple (**Apartado 5.6.5.2**).

Figura 91. Reconocimiento de color en tarjeta cognitiva

5.7.4 Comunicación

El ámbito cognitivo de la comunicación o lenguaje, abarca la capacidad de comprensión, escritura, lectura y fluidez verbal. En el desarrollo de este proyecto únicamente se realizan ejercicios terapéuticos centrados en la capacidad lectora, como se muestra a continuación.

5.7.4.1 Lectura

El diseño de las tarjetas encargadas de ejercitar la mecánica lectora del paciente se basan en el sistema de doble enunciado (**Apartado 5.6.5.3**). El primero indica la acción a realizar, generalmente leer en voz alta el texto indicado en el segundo enunciado, como puede observarse en la Figura 92.

Figura 92. Tarjeta de lectura como ámbito cognitivo

5.7.5 Capacidad ejecutiva

Los ejercicios terapéuticos enfocados a funciones ejecutivas se centran en la estimación temporal de acciones cotidianas, al razonamiento o a la resolución de problemas. En la realización de este proyecto se trata únicamente la estimación del tiempo.

5.7.5.1 Estimación temporal

La estimación temporal de funciones ejecutivas se realiza mediante el diseño de tarjetas que pregunten al paciente por la duración concreta de una acción, como es el tiempo necesario para hervir un café, o la duración de periodos como por ejemplo, la duración en meses de un año (Figura 93).

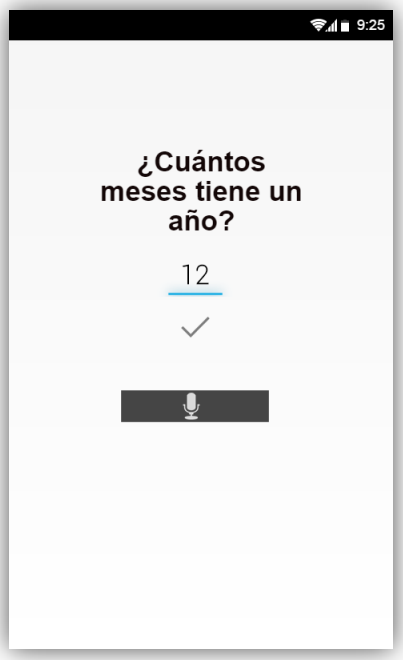


Figura 93. Diseño de tarjeta para la estimación temporal

5.7.6 Aritmética

El diseño de la terapia cuyo objetivo es mejorar la capacidad para realizar operaciones aritméticas, se realiza para favorecer el desempeño de tareas cotidianas para el paciente como es realizar la compra.

5.7.6.1 Cálculo

La terapia de cálculo involucra ejercicios de multiplicación, suma, resta y lectura de dígitos. Todos ellos utilizan tarjetas de doble enunciado (**Apartado 5.3.5.3**), el primero con las instrucciones para realizar el ejercicio y el segundo con las cifras y operadores implicados.

Tabla de multiplicar

Los ejercicios de multiplicación se basan en tarjetas que muestran la operación aritmética al paciente, quién debe responder con la cifra resultante. En la Figura 94 se observa un ejemplo de operación sobre el diseño de este tipo de tarjeta.

Figura 94. Tarjeta cognitiva para tabla de multiplicar

Lectura de números

Los ejercicios de lectura de números se realizan indicando en pantalla la cifra a leer por el paciente, como muestra la Figura 95.

Figura 95. Diseño de tarjeta para lectura de números

Operaciones de suma y resta

Los ejercicios de suma y resta utilizan tarjetas que presenten la operación aritmética al paciente, quién debe responder con la cifra resultante. La Figura 96 muestra el diseño con un ejemplo de operación suma.

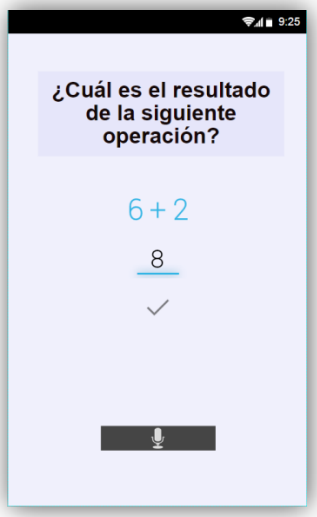


Figura 96. Tarjeta cognitiva para operaciones de suma y resta de un dígito

5.7.7 Gestión del dinero

La utilización del dinero supone una acción cotidiana realizada en cada actividad de compra y venta. En este proceso interviene tanto el cálculo mental como el reconocimiento de los dígitos. Por ello, el diseño de estas tarjetas debe mostrar una imagen donde se muestren las monedas y/o billetes involucrados en el ejercicio, como la mostrada en la Figura 97.

Figura 97. Gestión del dinero en tarjeta cognitiva

5.8 Adaptabilidad de la terapia

El paciente recibe las tarjetas cognitivas por lotes de dificultad, se presentarán en primera instancia las tarjetas de dificultad 1 hasta completar todas las disponibles. Al ser el primer lote no existirá adaptabilidad, una vez completado, se comprobará el índice de acierto de cada ámbito cognitivo.

En función del porcentaje obtenido se realizará la acción mostrada en la Tabla 15. Si ese porcentaje se encuentra entre un 0 y un 29% para un ámbito concreto, ya no se le presentarán más tarjetas de ese tipo en niveles sucesivos, al considerar que el grado de su enfermedad le impide realizarlos de forma satisfactoria. Si el índice de error se encuentra entre un 30% y un 70% será necesario reforzar dicho ámbito, para ello, al recuperar las tarjetas por dificultad se realiza una ordenación que agrupa las pertenecientes al ámbito cognitivo a reforzar, de esa manera se presentan al principio y de forma correlativa. Para un porcentaje mayor al 70% se considera una buena interacción con los ejercicios de ese ámbito por lo que se continúa trabajando sobre él.

Porcentaje de acierto	Acción
0 – 29 %	Eliminar el ámbito cognitivo de la terapia.
30 – 70%	Realizar un refuerzo positivo de ese ámbito cognitivo.
> 70%	Mantener el ámbito cognitivo en el siguiente nivel.

Tabla 15. Acciones realizadas en el proceso de adaptabilidad en función del índice de acierto

5.9 Estimación del grado de enfermedad

El método de estimación del estadio de Alzheimer en el que se encuentra el paciente que utiliza el sistema **HealthCoach,** se diseña a partir de la correlación de la escala de diagnóstico GDS, que divide en siete fases el deterioro cognitivo del paciente, con el BCRS, encargado de medir el deterioro de ámbitos cognitivos concretos. Dicha correlación es mostrada en la Tabla 13 del **Apartado 2.6.2**.

A partir de los tipos de tarjetas diseñadas en el **Apartado 5.7**, que contemplan todos los ámbitos cognitivos de la tabla anterior excepto el de praxis constructiva (implicaría por ejemplo desarrollar una interfaz que permita al usuario dibujar), se implementan tarjetas que representen los ejercicios presentados en la tabla de correlación. Como puede observarse en ella, son necesarios seis niveles de dificultad para las tarjetas en lugar de siete, ya que uno de ellos representa la ausencia de déficit.

El cálculo del estadio de la enfermedad en la aplicación de pacientes se realiza una vez el usuario ha terminado la terapia, bien porque ha realizado todas las tarjetas disponibles o porque el algoritmo de adaptabilidad descrito en el apartado anterior determina que ya no existen ámbitos cognitivos disponibles. Una vez calculado de forma transparente al usuario se envía el resultado por correo electrónico al responsable. Por su parte, el responsable también puede lanzar el proceso de cálculo desde la aplicación *HealthCoach Admin*.

Capítulo 6

Desarrollo

6.1 Funcionalidades generales

En este apartado, se describe el desarrollo de las funcionalidades generales usadas de forma reiterada en las dos aplicaciones Android, como son el uso de llamadas asíncronas por medio de la clase AsyncTask, uso de la librería JavaMail para envío de correos electrónicos e implementación del proceso de reconocimiento de voz y síntesis de texto a voz. También detalla el desarrollo e invocación de la API REST, utilizada para el intercambio de información de las aplicaciones cliente Android con la base de datos alojado en la parte servidor. Por último, se explica la implementación del código necesario para comprobar el estado de la conexión de red del dispositivo móvil, así como la interfaz de usuario.

6.1.1 Ejecución asíncrona con AsyncTask

La implementación de todas las tareas que requieran de un tiempo de ejecución considerable en las aplicaciones Android desarrolladas, deben ejecutarse de manera asíncrona por medio de la clase Android *AsyncTask*. Si no se realizara de esta manera y el hilo principal de ejecución quedara bloqueado durante un tiempo superior al establecido por Android, se mostraría en pantalla el mensaje de error *Application Not Responding*, lo que provocaría el cierre de la aplicación por parte del sistema.

Para ejecutar cualquier operación por medio de AsyncTask es necesario crear una nueva clase que extienda de *AsyncTask* y sobrescriba sus métodos:

- ***onPreExecute()*** – Este método se ejecuta antes del código principal que se ejecutará de forma asíncrona. Es útil para realizar todas las operaciones previas como la inicialización de mensajes informativos sobre el progreso de la operación.

- ***doInBackground()*** – El método más importante de las clases *AsynTask*, donde se realizan las operaciones en segundo plano. Hablando de este proyecto, en él se inicializan los componentes que realizan la petición HTTP, indicando la URL del servicio y el tipo de petición, y el posterior tratamiento del objeto JSON recuperado para extraer la información. Los **Apartados 6.1.3** y **6.1.4** explican en profundidad el proceso de comunicación con el servicio Web remoto.

- ***onPostExecute()*** – Es llamado de forma automática cuando finalizan las acciones llevadas a cabo en el método *doInBackground()*. Su invocación representa el momento idóneo para cerrar procesos abiertos o dar lugar a nuevos eventos.

Un ejemplo de definición de clase que hace uso de AsyncTask se muestra en la Figura 98.

```
private class GetUsers extends AsyncTask<Void, Void, Void> {

    @Override
    protected void onPreExecute() {
          super.onPreExecute();
    }

    @Override
    protected Void doInBackground(Void... arg0) {
    }

    @Override
    protected void onPostExecute(Void result) {
             super.onPostExecute();

    }
}
```

Figura 98. Estructura de clase que extiende AsyncTask

6.1.2 Implementación de la API REST

En la implementación del Servicio WEB REST, que interactúa con la base de datos del sistema, se da soporte a llamadas realizadas a través de cuatro **métodos HTTP**:

- **GET:** Para obtener datos a partir de un criterio de búsqueda especificado.
- **POST:** Utilizado para realizar inserciones de nuevos datos.
- **PUT:** Utilizado para actualizar datos existentes.
- **DELETE:** Permite borrar datos del sistema HealthCoach.

La figura 99 ilustra los elementos involucrados en el desarrollo o uso de la API REST.

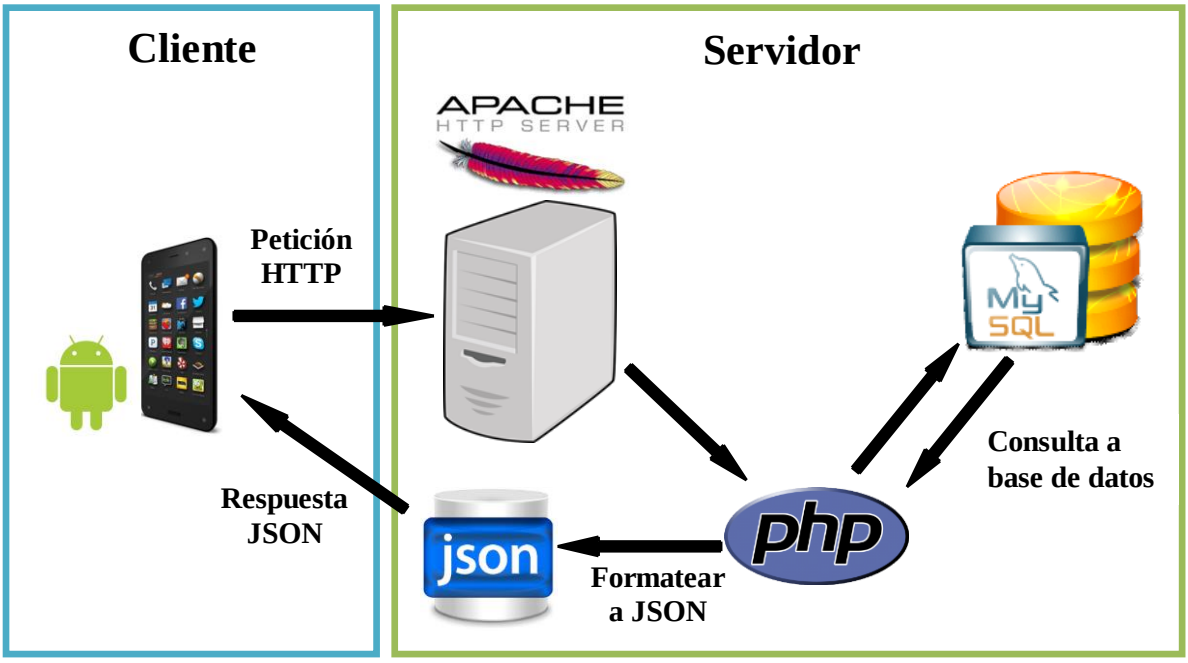


Figura 99. Diagrama de comunicación con base de datos a través de servicio WEB PHP

Toda la implementación se lleva a cabo en el fichero ***index.php*** alojado en la ruta */var/www/api* del servidor Apache. Junto a este fichero se encuentra la carpeta Slim que contiene la librería del mismo nombre, previamente instalada como explica el **Apartado 3.1.9** de esta memoria. Gracias a Slim se facilita el desarrollo de esta API, al aportar funciones de alto nivel que crean una capa de abstracción para trabajar con operaciones HTTP.

Adicionalmente, se ha definido un fichero ***.htaccess*** dentro de la carpeta del proyecto */api* para indicar a Apache la redirección necesaria para evitar especificar el fichero index.php en la URL que invoca a la API REST. Este fichero contiene las sentencias mostradas en la Figura 100.

```
RewriteEngine On

RewriteBase /api
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^(.*)$ index.php [QSA,L]
```

Figura 100. Contenido del fichero .htaccess

En cuanto al contenido del fichero *index.php*, comienza con el código necesario para incluir la librería Slim, invocar a la carga automática de dependencias e instanciar la aplicación, como puede observarse en la Figura 101.

```
require 'Slim/Slim.php';
\Slim\Slim::registerAutoloader();
$app = new \Slim\Slim();
```

Figura 101. Inicialización de Slim dentro del fichero index.php

Para establecer la conexión con la base de datos MySQL por medio de código PHP, se crea la función *getConnection()* que utiliza una instancia PDO para simplificar el proceso. Como muestra la Figura 102, basta con instanciar el objeto con la información necesaria para identificarse en la base de datos, como son la dirección IP del servidor donde se encuentra, el nombre de la base de datos, usuario y contraseña, como puede observarse en el código mostrado debajo. Además, se establece el juego de caracteres a UTF-8 para obtener los datos en el formato correcto. La última operación de la función retorna el objeto con la conexión creada.

```
function getConnection() {
    try {
        $db_username = "admin";
        $db_password = "introducir aquí la contraseña";
        $conn   =   new   PDO('mysql:host=192.168.1.19;dbname=proyectofincarrera',
$db_username, $db_password);
        $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $conn->exec("set names utf8");
    } catch(PDOException $e) {
        echo 'ERROR: ' . $e->getMessage();
    }
    return $conn;
}
```

Figura 102. Función PHP que establece conexión con la base de datos

El siguiente paso consiste en definir los **métodos de la API** donde se realizarán las operaciones con la base de datos.

- **getUsers**
- **getUser**
- **getUsersByImei**
- **getUsersAdminByImei**
- **getNumberUsersByAdmin**
- **getUsersByAdmin**
- **findByName**
- **getRecodsByUser**
- **getCards**
- **getcognitiveStats**
- **getTotalCognitiveStats**
- **getTotalCognitiveStatsBis**
- **getUserLevelCognitiveStats**
- **getDegree**
- **addUser**
- **addRecord**
- **updateUser**
- **updateDegree**
- **deleteUser**
- **deleteRecord**

Estos métodos (Figura 103) se utilizan en la asociación con las **rutas HTTP**, es decir, es necesario establecer que función será invocada en función del método HTTP recibido (GET, POST, PUT o DELETE) y de la URL solicitada.

```
$app->get('/users', 'getUsers');
$app->get('/users/:id',    'getUser');
$app->get('/usersImei/:id', 'getUsersByImei');
$app->get('/usersAdminImei/:id', 'getUsersAdminByImei');
$app->get('/numberUsersByAdmin/:id', 'getNumberUsersByAdmin');
$app->get('/usersByAdmin/:id', 'getUsersByAdmin');
$app->get('/users/search/:query', 'findByName');
$app->get('/records/:id','getRecordsByUser');
$app->get('/cards/:id/:level/:language', 'getCards');
$app->get('/cognitiveStats/:id', 'getCognitiveStats');
$app->get('/totalCognitiveStats/:id', 'getTotalCognitiveStats');
$app->get('/totalCognitiveStats', 'getTotalCognitiveStatsBis');
$app->get('/userLevelCognitiveStats/:id/:level', 'getUserLevelCognitiveStats');
$app->get('/degree/:id','getDegree');

$app->post('/users', 'addUser');
$app->post('/records', 'addRecord');

$app->put('/users/:id', 'updateUser');
$app->put('/degree/:id', 'updateDegree');

$app->delete('/users/:id',    'deleteUser');
$app->delete('/records/:id', 'deleteRecord');

$app->run();
```

Figura 103. Definición de rutas en la API REST

Una vez definida cada ruta y la función a la que invocará, se procede a arrancar la aplicación Slim con la sentencia *$app->run()*.

Debido al número de funciones desarrolladas y para no extender en exceso este apartado, se explicará el funcionamiento de una función de cada tipo de petición HTTP.

Petición GET - getUsersByImei($id)

Esta función devuelve en formato JSON todos los pacientes registrados con un determinado código IMEI indicado por parámetro. La definición de la ruta asociada a esta función es:

```
$app->get('/usersImei/:id', 'getUsersByImei');
```

Para llamar a esta API se utiliza la URL http://192.168.1.19:8080/api/usersImei/imei, sustituyendo "imei" por el valor real del código a consultar.

Entrando a explicar en detalle la implementación mostrada en la Figura 104, se observa la definición de la consulta SQL en la que se esperan dos variables, *id* relativo al IMEI y *type* referente al tipo de usuario (paciente o responsable). Una vez recuperada la conexión con la función *getConnection()* y guardada en el objeto *dbCon*, se utiliza el método *prepare* para asociarle la consulta SQL. Aún faltaría establecer el valor de las variables SQL, para ello se utiliza la sentencia $stmt->bindParam ("nombre del parámetro", valor). Para realizar la consulta se utiliza la sentencia PDO *execute()* y a continuación se crea el array *users* con las columnas de base de datos transformadas a objetos PDO utilizando el método *fetchAll(PDO::FETCH_OBJ)*. Después se cierra la conexión estableciendo a null el objeto dbCon y se genera la respuesta en formato JSON mediante la función Slim *json_encode($users)*, encargada de transformar a JSON el array pasado por parámetro.

De forma adicional, se ha encerrado el código en un bloque try-catch para capturar las posibles excepciones generadas en el código y poder generar un mensaje de error que será enviado en la respuesta a la petición realizada por el cliente.

```
function getUsersByImei($id) {
    $sql="SELECT
`IDU`,`NAME`,`SURNAME1`,`EMAIL`,`IMEI`,`LEVEL`,`GENDER`,`TYPE`,`LANGUAGE`     FROM
USERS WHERE IMEI=:id AND TYPE=:typ";
    try {
        $dbCon = getConnection();
        $stmt = $dbCon->prepare($sql);
        $type='P';
        $stmt->bindParam("id", $id);
        $stmt->bindParam("typ",$type);
        $stmt->execute();
        $users = $stmt->fetchAll(PDO::FETCH_OBJ);
        $dbCon = null;
        echo '{"users": ' . json_encode($users) . '}';
    } catch(PDOException $e) {
        echo '{"error":{"text":'. $e->getMessage() .'}}';
    }
}
```

Figura 104. Implementación de función getUsersByImei

A modo de ejemplo, en la Figura 105 se muestra la respuesta JSON generada en una invocación real a esta API. La petición GET se ha realizado con la URL: http://192.168.1.19:8080/api/usersImei/865092023515644

```
{
"users": [
          {
          "IDU": "34",
          "NAME": "Dave",
          "EMAIL": "admin@admin.es",
          "IMEI": "865092023515644",
          "LEVEL": "2",
          "GENDER": "M",
          "TYPE": "P",
          "LANGUAGE": "ENG"
          },
          {
          "IDU": "29",
          "NAME": "Iria",
          "EMAIL": "admin@admin.es",
          "IMEI": "865092023515644",
          "LEVEL": "1",
          "GENDER": "F",
          "TYPE": "P",
          "LANGUAGE": "ESP"
          }
          ]
}
```

Figura 105. Ejemplo de respuesta JSON con los usuarios registrados con un determinado IMEI

Petición POST - addRecord()

Esta función introduce en base de datos, a partir de una petición POST, un registro relativo a la respuesta que un paciente proporciona a una tarjeta de la terapia cognitiva. La definición de la ruta asociada a esta función es:

```
$app->post('/records', 'addRecord');
```

La URL utilizada para llamar a esta API es http://192.168.1.19:8080/api/records.

En cuanto a la implementación de la función, la interacción con la base de datos se realiza de igual manera que la petición GET explicada en la función *getUsersByImei()*. La diferencia radica en que los valores de los parámetros que se introducen en la consulta SQL viajan en el cuerpo de la petición POST, por lo que deben ser recuperados utilizando el método *request()* para obtener el objeto con la petición y a continuación recuperar los parámetros usando la sentencia *params('nombre del parámetro')*. En el código mostrado en la Figura 106, puede observarse como se recuperan los parámetros:

- **idc** – Es el identificador de la tarjeta (CARD) que se ha contestado.
- **response** – Es la respuesta proporcionada por el usuario y captada por el reconocedor de voz.
- **idu** – Es el identificador de usuario que ha realizado la respuesta.
- **karma** – Indica si ha sido una respuesta correcta o errónea, estableciendo el valor “S” o “N” respectivamente.

Otra diferencia en comparación a la petición GET anteriormente descrita radica en que una vez ejecutada la sentencia de inserción SQL ya no es necesario recuperar ningún dato, por lo tanto se suprime la llamada a *fetchAll()*. Únicamente se genera una respuesta informando del éxito de la operación y los parámetros introducidos.

```
function addRecord() {
    global $app;
    $req = $app->request(); // Getting parameter with names
    $paramIdc = $req->params('idc'); // Getting parameter with names
    $paramResponse = $req->params('response'); // Getting parameter with names
    $paramIdu = $req->params('idu'); // Getting parameter with names
    $paramKarma = $req->params('karma'); // Getting parameter with names

    $sql = "INSERT INTO RECORDS (`CREATED`,`IDU`,`IDC`,`RESPONSE`,`KARMA`) VALUES
(NOW(),:idu, :idc, :response, :karma)";
    try {
        $dbCon = getConnection();
        $stmt = $dbCon->prepare($sql);
        $stmt->bindParam("idu", $paramIdu);
        $stmt->bindParam("idc", $paramIdc);
        $stmt->bindParam("response", $paramResponse);
        $stmt->bindParam("karma", $paramKarma);
        $stmt->execute();
        $dbCon = null;
        echo   'Record   added!   Idc='.$paramIdc.'   Response='.$paramResponse.'
Idu='.$paramIdu.'  Karma='.$paramKarma;
    } catch(PDOException $e) {
        echo '{"error":{"text":'. $e->getMessage() .'}}';
    }
}
```

Figura 106. Implementación de la función addRecord()

Petición PUT - updateUser($id)

Esta función actualiza el nivel alcanzado en la terapia para poder proporcionarle tarjetas que se ajusten a ese parámetro. Es ejecutada al producirse una petición de tipo PUT a la URL http://192.168.1.19:8080 /api/users/id, debido al mapeo definido en:

```
$app->put('/users/:id', 'updateUser');
```

La implementación de la función *updateUser()* es similar a la realizada en la función *getUsersByImei()*, pero recuperando el parámetro *level* proporcionado en el cuerpo de la petición PUT y el *id* de usuario establecido en la propia URL. Después, ejecuta la sentencia UPDATE SQL:

```
$sql = "UPDATE USERS SET LEVEL=:level, MODIFIED=NOW() WHERE IDU=:id";
```

Petición DELETE - deleteRecord($id)

Esta función borra los registros con las respuestas de la terapia cognitiva de un paciente. Es ejecutada al producirse una petición de tipo DELETE con la URL http://192.168.1.19:8080/api/records/id, debido al mapeo definido en:

```
$app->delete('/records/:id', 'deleteRecord');
```

La implementación de la función *deleteRecord()* es similar a la realizada en la función *updateUser($id)*, recuperando el parámetro id que corresponde al IDU o identificador del usuario del que se van a borrar todos los registros. La sentencia SQL para la operación DELETE queda de la siguiente forma:

```
$sql = "DELETE FROM RECORDS WHERE IDU=:id";
```

6.1.3 Invocación a la API REST

Todas las peticiones de tipo GET o POST desde el código Android a la API REST se realizan siguiendo un mismo patrón. Son gestionadas por la clase *ServiceHandler*, de la que se destaca el método *makeServiceCall*. Este método recibe como parámetro la URL que identifica la API a invocar, el método HTTP (GET o POST) y la lista de parámetros enviados en el cuerpo de la petición.

Con la clase *DefaultHttpClient* se instancia un nuevo cliente HTTP, que ejecutará mediante el método execute() un objeto de tipo *HttpPost* o *HttpGet* previamente creado a partir de la URL del recurso, como muestra la Figura 107. La respuesta obtenida en la API externa se almacena en un objeto de tipo HttpResponse para transormarlo en String y devolverlo al final del método.

```
/*
 * Making service call
 * @url - url to make request
 * @method - http request method
 * @params - http request params
 * */
public String makeServiceCall(String url, int method,
                    List<NameValuePair> params) {
      try {
            // http client
            DefaultHttpClient httpClient = new DefaultHttpClient();
            HttpEntity httpEntity = null;
            HttpResponse httpResponse = null;

            // Checking http request method type
            if (method == POST) {
                  HttpPost httpPost = new HttpPost(url);
                  // adding post params
                  if (params != null) {
                        httpPost.setEntity(new UrlEncodedFormEntity(params));
                  }
```

```
                    httpResponse = httpClient.execute(httpPost);

              } else if (method == GET) {
                    // appending params to url
                    if (params != null) {
                          String paramString = URLEncodedUtils
                                      .format(params, "utf-8");
                          url += "?" + paramString;
                    }
                    HttpGet httpGet = new HttpGet(url);

                    httpResponse = httpClient.execute(httpGet);

              }
              httpEntity = httpResponse.getEntity();
              response = EntityUtils.toString(httpEntity);

        } catch (UnsupportedEncodingException e) {
              e.printStackTrace();
        } catch (ClientProtocolException e) {
              e.printStackTrace();
        } catch (IOException e) {
              e.printStackTrace();
        }

        return response;

}
```

Figura 107. Implementación de método makeServiceCall para realizar peticiones GET o POST

6.1.4 Lectura de objetos JSON

Una vez recuperada la respuesta de la API REST en formato *String*, como muestra el apartado anterior, es necesario realizar una interpretación de la cadena de texto para extraer los datos e incorporarnos a objetos Java. Primero se instancia una nueva clase JSONObject pasando como parámetro a su constructor el String devuelto por el servicio web, de esa manera se transforma a objeto JSON. Después será necesario recorrer la estructura JSON alternando el uso de los métodos getJSONArray() y getJSONObject() indicando como parámetro el nombre del array u objeto JSON que desea recuperarse.

La Figura 109 muestra un caso real de extracción de los datos JSON (Figura 108) correspondientes a registros de respuestas de la terapia cognitiva de un usuario, donde TAG_RECORDS es una etiqueta que representa “records”, TAG_IDC es “IDC” y TAG_KARMA es “KARMA”. Dentro del bucle que recorre el array de objetos JSON *records* se obtiene cada uno de los atributos y se establecen en un nuevo objeto Java de tipo *Record* para añadirlo finalmente al Vector *records_degree*.

```
{
  "records": [
    {
      "IDC": "25",
      "KARMA": "S"
    },
    {
      "IDC": "26",
      "KARMA": "S"
    }
   ]
}
```

Figura 108. JSON de array de objetos records

```
records_degree = new Vector<Record>();

if (jsonStr != null) {
	try {
		JSONObject jsonObj = new JSONObject(jsonStr);

		// Getting JSON Array node
		records = jsonObj.getJSONArray(TAG_RECORDS);

		// looping through All Contacts
		for (int i = 0; i < records.length(); i++) {

			JSONObject r = records.getJSONObject(i);

			int idc = r.getInt(TAG_IDC);
			String karma = r.getString(TAG_KARMA);

			Record temp = new Record();
			temp.setIdc(idc);
			temp.setKarma(karma);

			records_degree.add(temp);

		}
	} catch (JSONException e) {
		e.printStackTrace();
	}
} else {
	log.e("ServiceHandler", "Error recuperando datos");
}
```

Figura 109. Implementación de extracción de información JSON

6.1.5 Envío de correos electrónicos mediante JavaMail-Android

La aplicación de pacientes HealthCoach, envía de forma transparente al usuario correos electrónicos a su responsable con el propósito de mantenerle informado en tiempo real sobre los progresos y resultados de la adaptabilidad en la terapia mediante tarjetas cognitivas.

Para implementar esta funcionalidad se ha recurrido a **JavaMail-Android**, una adaptación a Android de la librería para Java llamada JavaMail. La instalación se realiza accediendo al repositorio del proyecto localizado en el enlace: https://code.google.com/p/javamail-android/downloads/list y descargando tres librerías en formato .jar:

- **additionnal.jar**
- **mail.jar**
- **activation.jar**

Una vez descargadas en nuestro equipo se importarán al proyecto Android arrastrándolas a la carpeta *libs*, de esa manera se añadirán automáticamente al *Build Path* para permitir el proceso de compilación de sus clases.

Además de una dirección de destino y un mensaje, debe establecerse un servidor SMTP a través del cual se realice el envío del correo hacia el destinatario. Para simplificar el proceso se ha decidido recurrir a los servidores SMTP de Gmail. Sólo es necesario autenticarse con una cuenta de Google para obtener una instancia de sesión correspondiente a *javax.mail.Session*, como se muestra en la Figura 110.

```
private Session createSessionObject() {
     Properties properties = new Properties();
     properties.put("mail.smtp.auth", "true");
     properties.put("mail.smtp.starttls.enable", "true");
     properties.put("mail.smtp.host", "smtp.gmail.com");
     properties.put("mail.smtp.port", "587");

     return Session.getInstance(properties, new javax.mail.Authenticator() {
            protected PasswordAuthentication getPasswordAuthentication() {
                  return new PasswordAuthentication("cuenta@gmail.com",
"contraseña");
            }
     });
}
```

Figura 110. Proceso de creación de una javax.mail.Session

También es necesario definir un método que cree la estructura del correo electrónico a partir de la clase *javax.mail.Message* proporcionada por JavaMail-Android. El método, mostrado en la Figura 111, recibe como parámetros la dirección de correo destino, el asunto, el mensaje del cuerpo y un objeto *Session* que represente la sesión con el servidor SMTP remoto. Como requerimiento de la aplicación HealthCoach se ha definido el contenido del correo en formato *text/html* para permitir el envío de correos HTML.

```
private Message createMessage(String email, String subject, String messageBody,
Session session) throws MessagingException, UnsupportedEncodingException {
          Message message = new MimeMessage(session);
          message.setFrom(new InternetAddress("HealthCoach@dlm.com", "HealthCoach
Admin"));
          message.addRecipient(Message.RecipientType.TO,new
InternetAddress(email, email));
          message.setSubject(subject);
          message.setContent(messageBody, "text/html");
          return message;
}
```

Figura 111. Método para la creación del correo electrónico mediante JavaMail-Android

Definidos los métodos que establecen la sesión SMTP y crean el correo electrónico, únicamente queda implementar su envío. Para ello, se crea una clase que extiende de AsyncTask y realiza en su método *doInBackground()* una llamada al método *send()* de la clase *Transport*, pasando como parámetro el objeto tipo *Message*:

```
Transport.send(messages[0]);
```

6.1.6 Reconocimiento de voz

Como ya se adelantó en el **Apartado 2.4**, Android facilita enormemente la implementación del reconocimiento de voz en las aplicaciones de terceros, permitiendo llamar a la aplicación de búsqueda por voz de Google instalada en la práctica totalidad de dispositivos por medio de un *Intent* definido a partir de la clase *android.speech.RecognizerIntent*. Una vez lanzado la aplicación de Google muestra una ventana con el mensaje "Habla ahora" (Figura 112) a la espera de reconocer los sonidos producidos por el habla del usuario.

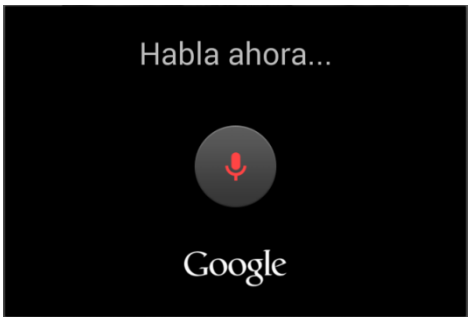


Figura 112. Pantalla de acceso oral a la búsqueda por voz de Google

La configuración y gestión del objeto *RecognizerIntent* se realiza a través de las etiquetas definidas en la propia clase, como muestra la Figura 113.

En HealthCoach, el *RecognizerIntent* se lanza desde el método *onClick()* que se ejecuta al pulsar el botón mostrado al usuario para responder al ejercicio presentado en cada tarjeta. En primer lugar se instancia el objeto Intent estableciendo su tipo a *RecognizerIntent.ACTION_RECOGNIZE_SPEECH*,

etiqueta que representa la actividad *android.speech.action.RECOGNIZE_SPEECH*. A este Intent se le añaden una serie de parámetros:

- **EXTRA_LANGUAGE_MODEL**. Establece el modelo de lenguaje que utilizará el reconocedor para optimizar la generación de resultados. Para este proyecto se ha seleccionado *LANGUAGE_MODEL_FREE_FORM* al ser el modelo optimizado para dictados de voz.
- **EXTRA_PROMPT.** Establece el texto mostrado en el cuadro de diálogo**.** Se ha elegido el valor "Habla ahora…".

Con el objetivo de lanzar la actividad de reconocimiento de voz, se utiliza el método *startActivityForResult(Intent intent, int requestCode)*, pasando como parámetros el propio objeto *Intent* y un código que identifica la petición para poder referenciarla posteriormente.

```
public void onClick(View v) {
     Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
     intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
     RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
     intent.putExtra(RecognizerIntent.EXTRA_PROMPT, "Habla ahora...");
     startActivityForResult(intent, REQUEST_CODE);
}
```

Figura 113. Invocación al RecognizerIntent de Android

Para recibir los resultados generados en el reconocedor es necesario sobrescribir el método *onActivityResult(int requestCode, int resultCode, Intent data)* proporcionado por Android para recibir la respuesta de un *Activity* a una petición lanzada a través de *Intent*. La Activity del reconocedor invoca este método con el código de petición que le invocó, un código de resultado y un objeto Intent con los datos generados en la respuesta. Ya dentro del método se invoca a la clase padre identificada en Java por *super* para indicar el final de la llamada a la actividad, después comprueba que el código de petición *requestCode* coincide con el pasado en el *startActivityForResult()* y que el código de respuesta posee el valor *RESULT_OK*. En caso afirmativo, se procesan los datos devueltos utilizando el método *getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS)* sobre el objeto data para obtener un array de tipo *String* con todas las cadenas de texto resultado del reconocedor de voz. La Figura 114 muestra un ejemplo de implementación.

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {

      super.onActivityResult(requestCode, resultCode, data);

       if (requestCode == REQUEST_CODE && resultCode ==Activity.RESULT_OK) {

       List<String> matches=
data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
        .
        .
        .
}
```

Figura 114. Obtención de los resultados del reconocedor de voz de Android

6.1.7 Síntesis de texto a voz

Para implementar la funcionalidad de síntesis de texto a voz en las aplicaciones Android desarrolladas en este proyecto, se recurre a un motor de síntesis previamente instalado en el dispositivo móvil gracias al paquete *android.speech.tts*. En el **Apartado 2.5** se describen las clases que conforman ese paquete, además se muestra una comparativa de los principales motores de síntesis disponibles en Google Play, así como el proceso de activación y configuración de *Síntesis de Google*, motor elegido en el desarrollo del sistema HealthCoach.

La inicialización del motor de síntesis se realiza instanciando un objeto de la clase *TextToSpeech*, pasando a su constructor como parámetros tanto el contexto de la aplicación como la implementación del interfaz *OnInitListener()*, en concreto del método *onInit(int status)*. La sintaxis exacta puede comprobarse en la Figura 115. En el método *onInit* se define el lenguaje de salida del motor a través de un objeto *Locale* de Java, en el que se establece la región geográfica y el lenguaje a través de sus códigos ISO. Por ejemplo, para español de España se establece el objeto *Locale* mediante la sentencia *new Locale("es","ES")* y para establecerlo al motor se utiliza el método *setLanguage(loc)*.

```
private TextToSpeech tts;

tts = new TextToSpeech(context, new TextToSpeech.OnInitListener() {
	@Override
	public void onInit(int status) {
		Locale loc;
		if (card.getLanguage().equals("ESP"))
			loc = new Locale("es", "ES");
		else
			loc = new Locale("en", "US");
		if (status != TextToSpeech.ERROR) {
			tts.setLanguage(loc);
		}
	}
});
```

Figura 115. Inicialización del motor de síntesis de voz a través de la clase TextToSpeech

Para reproducir las cadenas de texto a través del motor de síntesis basta con pasar el utilizar el método *speak (String text, int queueMode, HashMap<String,String> params)* sobre el objeto *tts* de tipo *TextToSpeech* previamente inicializado (Figura 116). El parámetro *text* representa el texto a partir del cuál se generará la voz, *queueMode* simboliza la forma de gestionar la pila de peticiones al motor de síntesis, si se indica el valor *TextToSpeech.QUEUE_ADD* la petición será procesada una vez finalicen las anteriores, si se utiliza *TextToSpeech.QUEUE_FLUSH* se abortarán todas las petitiones previas y se ejecutará inmediatamente ésta. Por último, *params* indica los parámetros enviados en la petición, puediendo enviarse a null.

```
public void speakText(String mensaje) {

	Toast.makeText(context, mensaje, Toast.LENGTH_SHORT).show();
	tts.speak(mensaje, TextToSpeech.QUEUE_ADD, null);

}
```

Figura 116. Reproducción de cadena de texto mediante motor TTS

Una vez finalizada la síntesis de texto a voz es importante parar y destruir la instancia del motor de síntesis, ya que esta funcionalidad es compartida en Android por cualquier aplicación que necesite invocarla y podría interferir en el correcto funcionamiento. Los métodos de *TextToSpeech* necesarios para realizar estas acciones son *stop()* y *shutdown()* respectivamente. Ambos métodos pueden invocarse dentro del método *onDestroy()* del Activity donde se realiza la síntesis de voz, de esa manera, se ejecutarán una vez que la actividad va a ser destruida. La Figura 117 muestra la implementación de este método.

```
@Override
public void onDestroy(){
	if (tts != null) {
		tts.stop();
		tts.shutdown();
		}
	super.onDestroy();
}
```

Figura 117. Desconexión del motor de síntesis de voz

6.1.8 Estado de conexión de red

Para un correcto funcionamiento de las dos aplicaciones desarrolladas se requiere de una conexión permanente a internet desde el móvil Android, ya sea a través de conexión Wi-Fi o a través de datos móviles.

Por lo tanto, es necesario definir un método que compruebe la conectividad. Android pone a disposición de los desarrolladores la clase *ConnectivityManager*, que una vez inicializada permite obtener esta información a través del método *getActiveNetworkInfo()*. Si el objeto obtenido de tipo *NetworkInfo* es nulo o nos devuelve un estado de no conectado o no disponible, puede afirmarse que el dispositivo móvil no dispone de conexión a internet. El contenido del método *checkConn()* se muestra en la Figura 118.

```
public boolean checkConn() {

	ConnectivityManager conMgr = (ConnectivityManager)
	this.getSystemService(Context.CONNECTIVITY_SERVICE);
	NetworkInfo i = conMgr.getActiveNetworkInfo();

	if (i == null) {
		return false;
	} else {

		if (!i.isConnected()) {
			return false;
		}

		if (!i.isAvailable()) {
			return false;
		}
		return true;
	}
}
```

Figura 118. Chequeo de la conexión a internet en el dispositivo

De manera complementaria, si el método anteriormente mostrado devuelve un estado de conexión false, se presenta al usuario un mensaje mediante *AlertDialog* informando de esta situación y se le insta a navegar a la configuración Wifi de su dispositivo para habilitarla. En el evento *onClick* del dialogo se captura tanto la respuesta positiva como negativa del usuario, en caso positivo se crea un nuevo Intent para acceder al Activity definido en la constante *Settings.ACTION_WIFI_SETTINGS*, correspondiente a la pantalla de configuración Wi-Fi del dispositivo. Si por el contrario pulsa en "No" se sale de la aplicación mediante un *System.exit(0)*. El método implementado se muestra en la Figura 119.

```
private void alert() {

      AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);
      alertDialogBuilder.setTitle("Sin conexión");
      alertDialogBuilder.setMessage("Para utilizar esta aplicación es necesaria
una conexión a internet, ¿desea ir a la configuración WIFI de su
dispositivo?").setCancelable(false).setPositiveButton("Sí",new
DialogInterface.OnClickListener() {
             public void onClick(DialogInterface dialog,int id) {
                   dialog.cancel();
                   Intent intent=new Intent(Settings.ACTION_WIFI_SETTINGS);
                   startActivity(intent);
                   System.exit(0);

             }});

      alertDialogBuilder.setNegativeButton("No", new
DialogInterface.OnClickListener() {

            public void onClick(DialogInterface dialog, int which) {
                  System.exit(0);
            }
      });

      AlertDialog alertDialog = alertDialogBuilder.create();
      alertDialog.show();
}
```

Figura 119. AlertDialog para gestionar conexión Wi-Fi

6.1.9 Interfaz de usuario

La interfaz gráfica se implementa a través de los denominados *layouts*, ficheros de extensión .xml que definen el aspecto visual de los componentes gráficos asociados a cada *Activity* de la aplicación. El primer criterio a la hora de crear el layout es decidir la forma en la que se van a presentar en pantalla los componentes que lo conforman, es decir su distribución. Android define varios tipos de *layouts*, a continuación se definen los utilizados en este proyecto:

- **Linear Layout**: Presenta todos los elementos en una única fila o columna. Por tanto, define la dirección horizontal o vertical en la que se disponen los componentes.
- **Relative Layout**: Presenta los elementos en relación a otros elementos o al padre.

En el siguiente apartado de esta memoria, referido a los módulos que conforman la aplicación Android de HealthCoach, se muestra la manera en la que se asocia un *Layout* con su *Activity* mediante ejemplos reales de la aplicación desarrollada.

6.2 Módulos comunes de HealthCoach y HealthCoach Admin

Esta sección presenta los módulos desarrollados para las aplicaciones Android que han sido reutilizados en **HealthCoach** y **HealthCoach Admin**, debido a la existencia de necesidades funcionalidades compartidas en ambas aplicaciones. Si se compara uno de estos módulos en ambas aplicaciones pueden presentarse algunas diferencias, se deben a las adaptaciones necesarias para una integración consistente con los requerimientos y diseño.

En el **Anexo B** puede visualizarse el diseño final y una descripción funcional de cada uno de los módulos.

6.2.1 Módulo de identificación y registro

Este módulo supone la primera interacción del usuario con la aplicación y le permite identificarse en el sistema o registrarse si es la primera vez que accede. Se trata de una funcionalidad compartida en la aplicación de pacientes y responsables, ya que ambos necesitan disponer de un usuario registrado en el sistema para acceder a las funcionalidades existentes en ambas aplicaciones.

6.2.1.1 Identificación

A modo introductorio se presenta en la Figura 120 el *layout* utilizado para la representación de este submódulo y así explicar su estructura. Implementa una pantalla de tipo lineal (*LinearLayout*), que contiene un *TextView* para mostrar el título de la pantalla y a continuación una lista de usuarios registrados en el dispositivo móvil, representada a través de un elemento *ListView.* Más abajo existe otro *TextView* para mostrar un posible mensaje de error en caso de no recuperarse usuarios *y* en la parte inferior se añade un último *TextView* que representará el enlace a la pantalla de registro de nuevo usuario.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#FFFFFF"
    android:orientation="vertical" >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10px"
        android:gravity="center"
        android:text="@string/quien"
        android:textColor="#000000"
        android:textSize="50dip" />

    <View
        android:layout_width="fill_parent"
        android:layout_height="3dp"
        android:background="#000000" />

    <FrameLayout
        android:layout_width="fill_parent"
        android:layout_height="0dip"
        android:layout_weight="1" >

        <ListView
            android:id="@android:id/list"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:dividerHeight="1dp"
            android:drawSelectorOnTop="true"
            android:gravity="center"
            android:textColor="#000000" />

        <TextView
            android:id="@android:id/empty"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:text="@string/sin_usuarios" />
    </FrameLayout>

    <View
        android:layout_width="fill_parent"
        android:layout_height="3dp"
        android:background="#000000" />

    <TextView
        android:id="@+id/link_to_register"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dip"
        android:gravity="center"
        android:text="@string/cuenta"
        android:textColor="#000000"
        android:textSize="40dip" />

</LinearLayout>
```

Figura 120. Implementación del layout user_selector.xml

Para mostrar el layout anterior es necesario asociarlo a su *Activity*, en este caso la actividad *UserSelector* que extiende la clase de Android *ListActivity*. Esta acción se realiza siempre desde el método *onCreate()* de la actividad, específicamente a través del método *setContentView()*, mostrado en la Figura 121.

```
@Override
public void onCreate(Bundle savedInstanceState) {

	super.onCreate(savedInstanceState);

	if (!checkConn()){
		alert();
	}else{
		setContentView(R.layout.user_selector);

		// Para permitir el acceso a la red desde el hilo principal, en caso
		// contrario StrictMode lo rechazaría
		StrictMode.setThreadPolicy(new StrictMode.ThreadPolicy.Builder()
				.permitNetwork().build());

		lblGotoRegister=(TextView) findViewById(R.id.link_to_register);

		imei = getMyImeiOrId();

		usuarios = new Vector<String>();
		usuariosObjetos = new Vector<User>();

		lblGotoRegister.setOnClickListener(new OnClickListener() {

			public void onClick(View view) {
				navegarRegistro();
			}
		});

		try {
			// Calling async task to get json
			new GetUsers().execute();
		} catch (Exception e) {
			Log.e("HealthCoach", e.getMessage(), e);
		}

	}

}
```

Figura 121. Método onCreate() del Activity UserSelector

Como puede observarse, la asociación de los componentes gráficos del layout se realiza con el método *findViewById()*, de esa forma pueden mapearse en objetos Java para establecer o recuperar sus características o incluso asociar eventos como muestra el ejemplo anterior, mediante el método *setOnClickListener()*. Esto permite realizar una acción específica al clicar sobre el componente sobre el que se ha definido el evento.

El usuario se identifica de forma unívoca en el sistema a través del código IMEI del terminal, o en su defecto del identificador de Android, un numero de 64 bits mostrado en hexadecimal que es

generado aleatoriamente en el primer arranque del dispositivo y permanece inalterable a lo largo del tiempo excepto si se realiza un "*reset factory*" para dejar el terminal como recién salido de fábrica. Para obtener el IMEI del dispositivo se recurre a la clase *TelephonyManager*, en el caso del id de Android es necesario utilizar la clase *Secure*, ambas implementaciones se observan en la Figura 122.

```
private String getMyImeiOrId() {

	String result = "";
	TelephonyManager mTelephonyMgr;
	mTelephonyMgr = (TelephonyManager)
getSystemService(Context.TELEPHONY_SERVICE);
	result = mTelephonyMgr.getDeviceId();

	if (result.equals(""))
		result = Secure.getString(getContentResolver(),
				android.provider.Settings.Secure.ANDROID_ID);

	return result;
}
```

Figura 122. Obtención del código IMEI o código Android del dispositivo

Una vez se dispone del código que identifica el terminal de forma única, se recurre al servicio web PHP para recuperar de base de datos el listado en formato JSON de usuarios registrados anteriormente con ese dispositivo.

La petición al servidor remoto puede demorarse en el tiempo y bloquear el hilo de ejecución principal, debido al tiempo de procesamiento que requiere una conexión HTTP, una traducción de la petición en la API PHP alojada en el servidor, consulta a la base de datos MySQL, formateo a JSON de los datos recuperados y posterior recepción de la información generada, es necesaria la utilización de una llamada asíncrona por medio de la clase AsyncTask de Android, ya explicada en el **Apartado 6.1.1**.

El primer paso es crear una nueva clase que extienda de AsyncTask y sobrescriba sus métodos *onPreExecute()*, *doInBackground()* y *onPostExecute()*. En el *onPreExecute()* se aprovecha para realizar todas las operaciones anteriores a la llamada en segundo plano, en este caso se establece un mensaje de progreso de la operación mediante la clase *ProgressDialog* de Android, que permite establecer el mensaje mediante el método *setMessage()* y establecer la posibilidad de cancelarlo mediante el método *setCancelable()*, como puede verse en la Figura 123.

```
@Override
protected void onPreExecute() {
	super.onPreExecute();
	// Showing progress dialog
	pDialog = new ProgressDialog(UserSelector.this);
	pDialog.setMessage("Recuperando usuarios, espere por favor...");
	pDialog.setCancelable(false);
	pDialog.show();
}
```

Figura 123. Diálogo de progreso definido dentro del método onPreExecute() de la clase AsyncTask

En el método *doInBackground()*, mostrado en la Figura 124, se inicializan los componentes que realizan la petición HTTP en segundo plano, indicando la url del servicio y el tipo de petición, y el

posterior tratamiento del objeto JSON recuperado para obtener la información de los usuarios obtenidos en objetos de tipo *User*. En el **Apartado 6.1.3** se explica en profundidad el proceso de comunicación con el servicio web remoto.

```
@Override
protected Void doInBackground(Void... arg0) {
	// Creating service handler class instance
	ServiceHandler sh = new ServiceHandler();

	// Making a request to url and getting response
	String jsonStr = sh.makeServiceCall(url + imei, ServiceHandler.GET);
	Log.d("Response: ", "> " + jsonStr);

	if (jsonStr != null) {
		try {
			JSONObject jsonObj = new JSONObject(jsonStr);

			// Getting JSON Array node
			contacts = jsonObj.getJSONArray(TAG_USERS);

			// looping through All users
			for (int i = 0; i < contacts.length(); i++) {
				JSONObject c = contacts.getJSONObject(i);
				String name = c.getString(TAG_NAME);
				String surname = c.getString(TAG_SURNAME);
				String email = c.getString(TAG_EMAIL);
				String idu = c.getString(TAG_IDU);
				String imei = c.getString(TAG_IMEI);
				int level = c.getInt(TAG_LEVEL);
				String gender = c.getString(TAG_GENDER);
				String language = c.getString(TAG_LANGUAGE);

				User temp = new User();
				temp.setIdu(idu);
				temp.setEmail(email);
				temp.setName(name);
				temp.setSurname(surname);
				temp.setImei(imei);
				temp.setLevel(level);
				temp.setGender(gender);
				temp.setLanguage(language);
				usuariosObjetos.add(temp);
			}
		} catch (JSONException e) {
			e.printStackTrace();
		}
	} else {
		Log.e("ServiceHandler", "Couldn't get any data from the url");
	}

	return null;
}
```

Figura 124. Llamada en segundo plano a la API REST para recuperar usuarios del dispositivo

En el método *onPostExecute()* se procede a eliminar el diálogo de progreso y a mostrar los datos de usuarios recuperados, estableciendo los valores necesarios para inicializar el *ListAdapter* mediante el método *setListAdapter()*, que los muestra en forma de lista a través del componente *ListView* del

layout. Si no se recuperan datos se invoca a un método que realiza la navegación hacia la página de registro. Toda esta gestión puede observarse en la porción de código mostrada debajo.

```
protected void onPostExecute(Void result) {

	super.onPostExecute(result);
	// Dismiss the progress dialog
	if (pDialog.isShowing())
		pDialog.dismiss();
	/**	* Updating parsed JSON data into ListView * */
	// Si no hemos recuperado usuarios navegamos a la pantalla de registro
	if (usuariosObjetos.isEmpty()) {
		navegarRegistro();
	} else {
		setListAdapter(new MyAdapter(UserSelector.this, usuariosObjetos));
	}
}
```

Figura 125. Inicialización de la lista de usuarios recuperados

El adaptador que define el formato de la lista con los usuarios disponibles se implementa en una nueva clase denominada *MyAdapter* que extiende a su vez de la clase *BaseAdapter* de Android. Posee un constructor que recibe el vector de objetos *User* y una referencia al *Activity* padre. El método *getView()* (Figura 126) es el más significativo devuelve una vista en forma de objeto *View* por cada elemento que compondrá el *ListView* final, para ello recupera del vector de usuarios cada elemento para definir los *TextView* e *ImageView* que deben mostrarse dentro de cada vista.

```
public View getView(int position, View convertView, ViewGroup parent) {

		LayoutInflater inflater = actividad.getLayoutInflater();
		User u = (User)lista.get(position);
		View view = inflater.inflate(R.layout.text_list, null, true);
		TextView textView =(TextView)view.findViewById(R.id.text1);
		textView.setText(u.getName()+" "+u.getSurname());
		textView.setTextColor(Color.BLACK);
		TextView nivel =(TextView)view.findViewById(R.id.text2);
		nivel.setText("Nivel "+u.getLevel());
		ImageView imageView=(ImageView)view.findViewById(R.id.image)
		if (((User)lista.get(position)).getGender().equals("M")){
		     imageView.setImageResource(R.drawable.male4);
		}else{
		     imageView.setImageResource(R.drawable.userfemale);
		}
		ImageView
imageViewLanguage=(ImageView)view.findViewById(R.id.imageLang);

		if (((User)lista.get(position)).getLanguage().equals("ESP")){
		     imageViewLanguage.setImageResource(R.drawable.spain);
		}else{
		     imageViewLanguage.setImageResource(R.drawable.england);
		}
		return view;
}
```

Figura 126. Método *getView()* del adaptador que define el formato de la lista de usuarios

El denominador común es que cada objeto View sigue la estructura definida por el layout *text_list.xml*, mostrado en la Figura 127. El primer elemento *ImageView* muestra una imagen con la representación de una silueta de hombre o mujer en función del género del usuario. El segundo *ImageView* con identificador *imageLang*, hace referencia a la imagen de la bandera de España o Inglaterra en función del lenguaje seleccionado por el paciente en el registro. Debajo se establecen dos TextView, para mostrar el nombre de usuario y el nivel alcanzado en la terapia cognitiva, respectivamente. Este módulo es utilizado para seleccionar al paciente en ambas aplicaciones, sin embargo, en la aplicación HealthCoach Admin existe de forma adicional la identificación del propio responsable, muy similar a la descrita hasta ahora pero invocando otro método de la API REST que devuelve únicamente usuarios de tipo responsable asociados al dispositivo desde el que accede. Tampoco inicializa el ImageView utilizado para mostrar la bandera ni el TextView que muestra el nivel alcanzado en las tarjetas cognitivas.

```
<?xml version="1.0" encoding="UTF-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/RelativeLayout1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
     android:orientation="vertical"
     android:layout_gravity="center_vertical"
    android:background="@drawable/blog_rounded_edges" >

    <ImageView
        android:id="@+id/image"
        android:layout_width="200dp"
        android:layout_height="100dp"
        android:layout_centerHorizontal="true"
        android:src="@drawable/user" />

     <ImageView
        android:id="@+id/imageLang"
        android:layout_width="100dp"
        android:layout_height="50dp"
        android:layout_marginTop="20dp"
        android:layout_alignParentRight="true"/>

    <TextView
        android:id="@+id/text1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/image"
        android:layout_centerHorizontal="true"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textSize="40dp"
        android:textColor="#000000" />

    <TextView
        android:id="@+id/text2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/text1"
        android:layout_centerHorizontal="true"
        android:textColor="#000000"
        android:textAppearance="?android:attr/textAppearanceLarge"/>
</RelativeLayout>
```

Figura 127. Layout que define el diseño de los elementos de la lista de usuarios

Finalmente, desde el método *onCreate()* del Activity se realiza la invocación de la clase que realizará la invocación al servicio web remoto de forma asíncrona de la siguiente manera. Esta porción de código se muestra en la Figura 128.

```
try {
    // Calling async task to get json
    new GetUsers().execute();

} catch (Exception e) {
    Log.e("HealthCoaching", e.getMessage(), e);
}
```

Figura 128. Invocación de clase que implementa una tarea asíncrona en Android

Para completar la fase de identificación, el paciente debe seleccionar su usuario dentro de la lista presentada, por lo tanto es necesario recoger el evento mediante un *onListItemClick()* (Figura 129) que recogerá los datos del usuario seleccionado para ser utilizados en otras pantallas de la aplicación.

```
@Override
protected void onListItemClick(ListView listView, View view, int position, long
id) {
      super.onListItemClick(listView, view, position, id);
      User o = (User)getListAdapter().getItem(position);
      Intent i = new Intent(this, HiScreen.class);
      i.putExtra("user", o.getName());
      i.putExtra("surname", o.getSurname());
      i.putExtra("idu", o.getIdu());
      i.putExtra("level", o.getLevel());
      i.putExtra("emailAdmin", o.getEmail());
      i.putExtra("language", o.getLanguage());
      startActivity(i);
      this.finish();
}
```

Figura 129. Acción que captura la selección de un elemento dentro de la lista de usuarios

6.2.1.2 Registro

El submódulo de registro permite al usuario que no lo había realizado previamente, darse de alta en el sistema para poder acogerse al sistema de monitorización y adaptabilidad de la terapia cognitiva. Por lo tanto se compone de una serie de campos *EditText* en los que introducir los datos de usuario, destacando la peculiaridad del componente *Spinner* que representa un combo-box en Android. Este componente se utiliza para establecer tanto el género como el idioma de las tarjetas de terapia.

Cogiendo como ejemplo el registro de la aplicación HealthCoach, en primer lugar se asocia el objeto *Spinner* del layout a un objeto Java de la forma habitual, utilizando *findViewById(R.id.spinnerLanguage) e* indicando como parámetro el mismo identificador que tiene establecido en la vista. Después se crea un adaptador de tipo array de String mediante la clase *ArrayAdapter*, que necesitará como argumentos un *layout* que represente el formato de las filas *del Spinner* y un *Array* de tipo *String* con los valores que se mostrarán, en este caso “Inglés” y “Castellano”.

Una vez definido el objeto ArrayAdapter se establece al objeto Spinner, y se asocia un *listener* o escuchador para el evento de selección de cada elemento mostrado dentro del Spinner. Se utiliza el método *setOnItemSelectedListener()* de la clase Spinner, que recibe como parámetro un nuevo listener definido a partir de la clase *AdapterView* y la implementación del método *onItemSelected()*. Este método tiene como cometido realizar la asociación de posición numérica del elemento pulsado con la posición del Array de idiomas definidos. En la 130 se observa todo el procedimiento.

```
spnrLang = (Spinner) findViewById(R.id.spinnerLanguage);
ArrayAdapter<String> adapterLang = new ArrayAdapter<String>(this,
R.layout.spinner_layout, languages);
spnrLang.setAdapter(adapterLang);
spnrLang.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener(){
       @Override
       public void onItemSelected(AdapterView<?> arg0, View arg1, int arg2, long
arg3) {
           int position = spnrLang.getSelectedItemPosition();
           idiomaSeleccionado=languages[+position];
        }
});
```

Figura 130. Implementación de Spinner para selección de idioma

Una vez completados los campos y pulsado el botón “Enviar” se realiza una validación para comprobar que todos estén informados, en caso contrario se mostraría en pantalla un mensaje que insta a rellenar completamente el formulario.

6.2.2 Módulo de menú principal

Este módulo ejerce en la aplicación **HealthCoach** la doble labor de presentar al usuario recién identificado un mensaje de bienvenida con datos asociados a él como son el nombre y el nivel de dificultad alcanzado en los ejercicios cognitivos, y facilitar un menú con las opciones que puede realizar, como son comenzar los ejercicios de terapia cognitiva, consultar las estadísticas, cambiar de usuario o salir de la aplicación.

En **HealthCoach Admin** existe la misma funcionalidad dual con matices. Muestra el nombre y nivel en la terapia del paciente seleccionado, no los datos relativos al responsable identificado en el sistema. En cuanto al menú con las opciones disponibles para el responsable, no existe la opción de comenzar los ejercicios terapéuticos, en su lugar se presenta la posibilidad de consultar el grado estimado de Alzheimer para el paciente seleccionado.

La interfaz de este módulo en las aplicaciones de paciente y responsable puede consultarse en el **Anexo B**. Para implementar las funcionalidades de este módulo en Android se crea un *Activity* llamado *HiScreen* con un método *onCreate()* utilizado para obtener las referencias de los *TextView* del *layout* asociado y así poder establecer su valor de la forma habitual, mostrada en la Figura 131.

```
super.onCreate(savedInstanceState);
setContentView(R.layout.hi_screen);
usr_name = (TextView) findViewById(R.id.usr_name);
```

Figura 131. Asociación de un layout a su Activity

Para establecer el valor del *TextView* asociado al nombre de usuario, instanciado en el objeto *usr_name*, se recuperan los parámetros inyectados en la invocación a la actividad HiScreen mediante *Intent*. Más adelante se explica cómo se establecen estos parámetros ya que también se realiza al pulsar sobre una opción del menú de opciones, para navegar a la nueva actividad y pasar los datos necesarios en esa nueva pantalla. Ahora se explica el proceso de obtener estos parámetros dentro del método *onCreate*() de la actividad invocada.

Android pone a disposición mediante su SDK la clase ***Bundle***, capaz de almacenar todos los parámetros o "extras" establecidos en la llamada a una nueva Activity por medio de un Intent. Una vez inicializado el objeto Bundle como se muestra en la Figura X, pueden recuperarse mediante los métodos *getString("nombre_del_parámetro")* o *getInt("nombre_del parámetro")* los parámetros de tipo *String* o tipo *int* respectivamente (Figura 132).

```
Bundle extras = getIntent().getExtras();

// Obtenemos datos enviados en el Intent
if (extras != null) {
       user = extras.getString("user");
       surname = extras.getString("surname");
       idu = extras.getString("idu");
       level = extras.getInt("level");
       emailAdmin = extras.getString("emailAdmin");
       language = extras.getString("language");
}
```

Figura 132. Bundle como método para almacenar datos a través de los Activity

Una vez recuperado el valor del nombre de usuario, almacenado en el String user, puede establecerse en el TextView del layout mediante el método *setText(String)*:

```
usr_name.setText("Bienvenido "+user+"!");
```

Para habilitar la navegación de los elementos del menú, presentados mediante un TextView, se le añade un escuchador o *listener* de tipo *OnClickListener* que capture el evento de pulsación sobre el texto. Una vez capturado puede realizarse cualquier acción como la invocación a otro método. Como ejemplo, se muestra el código de la Figura 133 que establece al *TextView* de la opción de menú "Comenzar :-)" el listener necesario para ejecutar el método *lanzarTarjetas()* que iniciará el módulo de terapia cognitiva.

```
start.setOnClickListener(new View.OnClickListener() {

      public void onClick(View view) {
            lanzarTarjetas();
    }

});
```

Figura 133. Asociación de evento OnClickListener a objeto TextView

Por último, continuando con el ejemplo anterior, el método *lanzarTarjetas()* prepara el *Intent* que lanzará la actividad *ScreenSlidePagerActivity* encargada de mostrar las tarjetas de terapia en pantalla. Como puede comprobarse en la Figura 134, una vez inicializado el Intent utilizando su constructor que requiere como parámetros el contexto y el nombre de la clase de la Activity a la que se invoca, se utiliza el método *putExtra()* para establecer la dupla parámetro-valor. Sólo falta lanzar el Intent mediante el método *startActivity(intent)* que llevará el hilo principal de ejecución a la nueva actividad y finalizar la actividad actual mediante el método *finish()*, con el fin de liberar recursos.

```
public void lanzarTarjetas(){
      Intent i = new Intent(this, ScreenSlidePagerActivity.class);
      i.putExtra("user", user);
      i.putExtra("surname", surname);
      i.putExtra("idu", idu);
      i.putExtra("level", level);
      i.putExtra("emailAdmin", emailAdmin);
      i.putExtra("language", language);
      startActivity(i);
      this.finish();
}
```

Figura 134. Método para invocar a la actividad ScreenSlidePagerActivity

Por lo tanto, y a modo de resumen, gracias al método *putExtra()* de Intent se pueden establecer parámetros que serán recuperados en la actividad destino, mediante el uso de la clase *Bundle*.

6.2.3 Módulo de tarjetas cognitivas

El módulo encargado de gestionar la terapia mediante tarjetas cognitivas es exclusivo de la aplicación **HealthCoach**. Su implementación se realiza a través de dos clases, *ScreenSlidePageFragment* y *ScreenSlidePagerActivity*. Cada una de ellas se encarga de realizar las siguientes tareas:

- **ScreenSlidePagerActivity**
    - Realiza el proceso de **adaptabilidad** en la terapia decidiendo qué tipos de tarjetas cognitivas serán presentadas al paciente y en qué casos es necesario un refuerzo en la realización de un ámbito cognitivo concreto.
    - Recupera mediante conexión con la **API REST** la información de las tarjetas pertenecientes al nivel de dificultad en el que se encuentra el usuario.
    - Crea las tarjetas mediante **fragmentos** y el paginador que los contiene.
    - Gestiona la transición entre **tarjetas.**
    - Guarda en base de datos las **respuestas** a cada una de las tarjetas cognitivas, indicando si es correcta o errónea y la respuesta proporcionada mediante el habla. Utiliza la API REST para este propósito.
    - Realiza el cálculo estimado del **grado de enfermedad** del paciente.
    - Envía al responsable **correos informativos** con el resultado de la terapia cognitiva al finalizar las tarjetas de un nivel concreto y con el resultado de la estimación del grado de Alzheimer al completar suficientes tarjetas cognitivas.

- **ScreenSlidePageFragment**

    - Representa la **interfaz gráfica** de cada una de las tarjetas en función del ámbito cognitivo que representa, utilizando objetos de tipo ***Fragment***.
    - Gestiona el **reconocimiento de voz** para captar las respuestas del paciente y valida si son correctas.
    - Gestiona la **síntesis de texto a voz** para proporcionar información adicional al usuario.

Una vez esquematizadas todas las tareas, se procede a explicar por orden de presentación cada una de ellas.

6.2.3.1 Adaptabilidad

La implementación de la adaptabilidad persigue cumplir siempre la funcionalidad definida en el **Apartado 5.8** de forma transparente al usuario, en lugar de permitirle elegir el tipo de ejercicios a realizar, es la propia aplicación la que decida qué tipo de preguntas mostrar y en cuáles hacer hincapié.

Los pasos seguidos en el código desarrollado son:

1) Crear dos vectores, el primero *Vector<String> ambitosCognitivosVisibles* almacena elementos de tipo *String* con el nombre de los ámbitos cognitivos que estarán disponibles para las tarjetas del usuario en función de los resultados obtenidos para el nivel anterior, el segundo *Vector ambitosCognitivosOrdenados* guarda los ámbitos cognitivos que son necesarios reforzar en objetos de tipo *AmbitoCog*. Esta clase se crea con un atributo de tipo *String* que representa el nombre del ámbito y otro de tipo *int* con el porcentaje de acierto en ese ámbito cognitivo.

2) Crear seis variables de tipo *int* que almacenen de forma temporal el índice de acierto de cada ámbito cognitivo en el nivel indicado: *porcentajeMemoria*, *porcentajeGnosias*, *porcentajeOrientacion*, *porcentajeAritmetica*, *porcentajeCoumunicacion*, *porcentajeEjecutiva*.

3) Crear otras seis variables de tipo int para almacenar el número de tarjetas contestadas por el usuario de forma correcta en cada ámbito cognitivo del nivel indicado: *memoriaUser*, *orientacionUser*, *gnosiasUser*, *aritmeticaUser*, *comunicacionUser*, *ejecutivaUser*.

4) Crear seis variables de tipo *int* para almacenar el número total de tarjetas existentes por ámbito cognitivo para un nivel dado: *memoria*, *gnosias*, *orientacion*, *aritmetica*, *comunicacion*, *ejecutiva*.

5) Si el nivel de dificultad actual de las tarjetas es mayor que 1, significa que se ha superado la primera ronda y por lo tanto se procede a generar las estadísticas de acierto por ámbito cognitivo, en caso contrario se cancela el proceso.

6) Se obtiene el número total de tarjetas por ámbito cognitivo para el nivel de dificultad anterior al actual del usuario, realizando una petición de tipo GET a la API REST remota que ejecuta en código PHP la consulta a base de datos:

```
$sql = "SELECT `COGNITIVE`, COUNT(*) FROM CARDS WHERE DIFFICULTY=:level
GROUP BY `COGNITIVE`";
```

La petición al servicio Web lleva un parámetro en la url que indica el nivel a mapear en la variable *:level*. Una vez obtenida la respuesta en forma de objeto JSON, se establecen los valores de las variables definidas en 4).

7) Se obtiene número de tarjetas contestadas correctamente por ámbito cognitivo para el nivel de dificultad anterior al actual para el usuario, realizando una petición de tipo GET al servicio Web, que finalmente realiza la consulta a base de datos:

```
$sql = "SELECT `COGNITIVE`, COUNT(*) FROM CARDS WHERE IDC IN ( SELECT IDC
FROM RECORDS WHERE IDU=:id AND KARMA=:kar) AND DIFFICULTY=:level GROUP BY
`COGNITIVE`";
```

En este caso se utilizan dos parámetros en la URL, el primero para indicar el identificar de usuario y el segundo para establecer el nivel. Adicionalmente se establece el valor del KARMA, es decir si la respuesta a la tarjeta fue correcta o no, a "S" para obtener únicamente las respuestas correctas. Una vez recuperado el objeto JSON resultante se establecen los valores en las variables declaradas en el paso 3).

8) En este punto se procede a calcular el porcentaje de acierto de cada ámbito cognitivo gracias a los datos obtenidos en los pasos 6) y 7), si es mayor o igual del 70% se añade el nombre de ese ámbito al Vector *ambitosCognitivosVisibles y* si se encuentra entre el 30 y el 70% se añade tanto a éste como a *ambitosCognitivosOrdenados*, creando el objeto *AmbitoCog* con el nombre y porcentaje de acierto.

9) Por último, se ordena de menor a mayor la colección *ambitosCognitivosOrdenados* de ámbitos cognitivos a destacar en función del porcentaje de acierto, para incidir en la realización de los ámbitos cognitivos que más dificultad presentan al usuario. El código utilizado para ordenar el vector se muestra en la Figura 135.

```
Collections.sort(ambitosCognitivosOrdenados, new Comparator<AmbitoCog>(){

	@Override
	public int compare(AmbitoCog o1, AmbitoCog o2) {
		return new Integer(o1.getPorcentaje()).compareTo(new
Integer(o2.getPorcentaje()));
	}
});
```

Figura 135. Método de ordenación de vector de ámbitos por porcentaje de acierto creciente

6.2.3.2 Información de tarjetas

La información necesaria para generar las tarjetas de la terapia cognitiva se obtiene mediante invocación a la API remota. El procedimiento utilizado es el indicado en los **Apartados 6.1.1**, **6.1.2**, **6.1.3** y **6.1.4**, es decir, mediante la creación de una clase que extiende de *AsyncTask* llamada *GetCards* que realiza de forma asíncrona la llamada HTTP mostrada en la Figura 136.

```
private static String url = "http://192.168.1.19:8080/api/cards";
String jsonStr = sh.makeServiceCall(url+"/"+idu+"/"+level+"/"+language,
ServiceHandler.GET);
```

Figura 136. Información de tarjetas mediante invocación a API REST

La función PHP que se ejecuta en el servidor remoto realiza una consulta a base de datos para obtener la información que formará el objeto JSON. Los campos recuperados de la tabla CARDS son:

- **IDC** – Identificador único de la tarjeta.
- **COGNITIVE** – El ámbito cognitivo al que pertenece.
- **INSTRUCTIONS** – Representa el contenido del enunciado principal de la tarjeta.
- **INFO** – Enunciado complementario al principal.
- **RESPONSE** – Este campo almacena la respuesta correcta que deberá proporcionar el paciente para la tarjeta actual.
- **THEME** – Representa el subtipo de ejercicio asociado a un ámbito cognitivo. Todos los disponibles se encuentran en el **Apartado 5.7**.
- **DIFFICULTY** – Establece el nivel de dificultad de la tarjeta, comprendido entre 1 y 6.
- **IMAGE_URL** – Contiene la referencia a la imagen presentada en la tarjeta, en caso de ser necesaria.
- **LANGUAGE** – El idioma en el que se presenta la tarjeta, “ESP” para castellano y “ENG” para tarjetas en inglés.

La función PHP ejecutada se muestra en la Figura 137.

```
function getCards($id,$level,$language) {
$sql_query="select`IDC`,`COGNITIVE`,`INSTRUCTIONS`,`INFO`,`RESPONSE`,`THEME`,`DIFF
ICULTY`,`IMAGE_URL`,`LANGUAGE` FROM CARDS WHERE IDC NOT IN ( SELECT IDC FROM
RECORDS WHERE IDU=:id AND KARMA IN ('S','N')) AND LANGUAGE=:language AN$
    try {
        $prueba ='S';
        $dbCon = getConnection();
        $stmt = $dbCon->prepare($sql_query);
        $stmt->bindParam("id",$id);
        $stmt->bindParam("level",$level);
        $stmt->bindParam("language",$language);
        $stmt->execute();
        $cards  = $stmt->fetchAll(PDO::FETCH_OBJ);
        $dbCon = null;
        echo '{"cards": ' . json_encode($cards) . '}';
    } catch(PDOException $e) {
        echo '{"error":{"text":'. $e->getMessage() .'}}';
    }
}
```

Figura 137. Función PHP para obtener datos de tabla CARDS

Una vez obtenido el objeto JSON con los datos correspondientes a cada tarjeta, se recorre para crear las instancias de tipo *Card*, descartando las tarjetas cuyo ámbito cognitivo no se encuentre entre los permitidos por el procedo de adaptabilidad, almacenados en el vector *ambitosCognitivosVisibles*. Este proceso se realiza dentro del método *doInBackGround()* de la clase asíncrona que realiza la llamada a la API REST, con el código mostrado en la Figura 138.

```
Card temp = new Card();
temp.setId(idc);
temp.setInfo(info);
temp.setCognitive(cognitive);
temp.setResponse(response);
temp.setTheme(theme);
temp.setInstructions(instructions);
temp.setImage_url(imageUrl);
temp.setLanguage(cardLanguage);

//aplicamos la adaptabilidad
if (ambitosCognitivosVisibles.contains(temp.getCognitive()))
      prueba.add(temp);
```

Figura 138. Proceso de adaptabilidad con ámbitos cognitivos permitidos

Por lo tanto, al final se tiene el vector *prueba* que contiene únicamente las tarjetas de ámbitos cognitivos que el paciente debe realizar. Ahora falta ordenar este vector priorizando las tarjetas cuyo ámbito debe reforzarse, este proceso se realiza dentro del método *ordenarTarjetasAmbitosRefuerzo()*. A partir del contenido del vector *ambitosCognitivosOrdenados* descrito en el **Apartado 6.2.3.1**, se recorre por orden los ámbitos cuyo refuerzo es prioritario para buscar las tarjetas que pertenecen a esos ámbitos y añadirlos al nuevo vector llamado *tarjetasOrdenadas*. Al acabar los ámbitos con refuerzos se añade el resto de tarjetas que todavía existían en el vector prueba y que por lo tanto no requerían de refuerzo.

Sólo falta recorrer el vector *tarjetasOrdenadas* para recuperar cada una de las tarjetas (objetos Card) y crear nuevos fragmentos de tipo ScreenSlidePageFragment a partir de ellas. Cada uno de estos fragmentos se añaden al vector *fragments*.

Este proceso puede observarse en el código de la Figura 139.

```
ordenarTarjetasAmbitosRefuerzo();

for (int i = 0; i < tarjetasOrdenadas.size(); i++) {

      Card carta = (Card) tarjetasOrdenadas.elementAt(i);
       ScreenSlidePageFragment fragment = new    ScreenSlidePageFragment(carta,
this.getApplicationContext());

       fragments.add(fragment);
}
```

Figura 139. Ordenación del vector de tarjetas en función de los ámbitos a reforzar

En este punto, una vez obtenido el vector de fragments definitivo al pasar el proceso de adaptabilidad para el nivel de dificultad actual, puede procederse a realizar la asociación del ViewPager con el adaptador creado, descrito en el **Apartado 6.2.3.5**.

6.2.3.3 Tarjetas como *Fragments*

Para desarrollar la manera en la que se le presentan las tarjetas al usuario durante la terapia cognitiva, se ha recurrido a la clase *ViewPager* de Android, al proporcionar una vista que muestra un conjunto de páginas sobre las que desplazarse deslizando el dedo sobre la pantalla táctil de izquierda a derecha o viceversa. Estas páginas representarán nuestras tarjetas, denominadas así de ahora en adelante, que son implementadas mediante fragmentos (*Fragment*) de Android. Para realizar la conexión entre ***Fragment*** y ***ViewPager*** se utiliza el adaptador *FragmentStatePagerAdapter*.

Para utilizar *ViewPager* en la aplicación es necesario utilizar la librería de soporte de Android, denominada ***Support Library***. Esta librería permite utilizar características de una API superior al nivel de API seleccionado en el proyecto, de esa manera la aplicación disfrutar de características más modernas y seguir siendo compatible con **Android 1.6** (**API nivel 4**). Su instalación es tan sencilla como seleccionar el proyecto en Eclipse, clicar con el botón derecho del ratón y seleccionar *Android Tools* > *Add Support Library*. Esto añadirá de forma automática la librería **android-support-v4.jar** en el directorio /lib del proyecto y será introducida en el Build Path para poder ser utilizada en el proceso de compilación.

Generación automática de tarjetas

Para crear la vista que mostrará el contenido de cada tarjeta, es decir la que utilizará el *Fragment*, se implementan tres *layouts* para cubrir los tres tipos de pantalla descritos en el **Apartado 5.6.5**:

- ***fragment_tarjeta_caras.xml*** – Representa la vista para tarjetas que muestran imágenes. Se compone de un *LinearLayout* que contiene un *TextView* para mostrar el enunciado del ejercicio y un *ImageView* para representar la imagen que complementa al enunciado.
- ***fragment_tarjeta_detalle.xml*** – Utilizada para la vista de tarjetas con doble descripción, por lo tanto utiliza un doble *TextView*.
- ***fragment_tarjeta_dos.xml*** – Implementa la vista de las tarjetas que requieren un enunciado simple. Es la versión más sencilla requiriendo un único TextView para mostrar el enunciado.

De forma común, los tres tipos de vistas comparten los elementos existentes en la parte inferior de las tarjetas, donde se establece el botón para proceder a responder la tarjeta gracias a un *ImageButton* y la respuesta proporcionada por el usuario y devuelta por el reconocedor de voz, mediante un elemento de tipo *ListView*.

De cara a mostrar el código de una de las vistas, se ha elegido la correspondiente a *fragment_tarjeta_caras.xml* ya que contiene tanto los elementos *TextView* como *ImageView*. Se muestra en la Figura 140.

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/content"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    android:background="#FFFFFF"
    android:gravity="center" >
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:orientation="vertical"
        android:padding="15dp" >
        <TextView
            android:id="@+id/infoEncuesta"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:lineSpacingMultiplier="1.1"
            android:text="texto"
            android:textColor="#000000"
            android:textSize="45sp"
            android:textStyle="bold" />
        <ImageView
            android:id="@+id/imageViewId"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center" />
        <ImageButton
            android:id="@+id/recognize"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:contentDescription="boton voz"
            android:onClick="startVoiceRecognition"
            android:src="@drawable/voicebutton" />
        <ListView
            android:id="@+id/word_list"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:gravity="center"
            android:textColor="#000000" >
        </ListView>
    </LinearLayout>

</ScrollView>
```

Figura 140. Implementación de layout fragment_tarjeta_caras.xml

El siguiente paso consiste en crear la clase *ScreenSlidePageFragment* que hereda de *Fragment*. Para cada tarjeta será necesario crear una nueva instancia de esa clase, indicando como parámetros la clase *Card* que contiene todos los datos relativos a la tarjeta y el contexto (*Context*) de la aplicación. Cuando se dibuje en pantalla la tarjeta se ejecuta el método *onCreateView* donde se establecen los textos e imágenes definidos en cualquiera de los tres layouts descritos. En relación a ésto, se ha desarrollado un sistema de **selección automática del layout** en función del tipo de ejercicio que mostrará la tarjeta. Como muestra la Figura 141, gracias a la estructura definida en la instrucción *switch* se utiliza el valor del parámetro *type*, que representa el tipo de tarjeta, para establecer el layout utilizado así como inicializar sus componentes.

```
View v;
image=card.getImage_url();

switch (type) {
      case 'E':
      v = inflater.inflate(R.layout.fragment_tarjeta_caras, container,
false);
      mImageView = (ImageView) v.findViewById(R.id.imageViewId);
      mImageView.setImageResource(getResources().getIdentifier(image,
"drawable", context.getPackageName()));
      wordListView = (ListView) v.findViewById(R.id.word_list);
      btnRecognize = (ImageButton) v.findViewById(R.id.recognize);
      btnRecognize.setOnClickListener(this);
      break;
       case 'M':
      v = inflater.inflate(R.layout.fragment_tarjeta_dos,
container,false);
      wordListView = (ListView) v.findViewById(R.id.word_list_dos);
      btnRecognize2 = (ImageButton) v.findViewById(R.id.recognize2);
      btnRecognize2.setOnClickListener(this);
      break;
       case 'T':
      v =inflater.inflate(R.layout.fragment_tarjeta_caras, container,
false);
      wordListView = (ListView) v.findViewById(R.id.word_list);
      mImageView = (ImageView) v.findViewById(R.id.imageViewId);
      mImageView.setImageResource(getResources().getIdentifier(image,
"drawable", context.getPackageName()));
      btnRecognize = (ImageButton) v.findViewById(R.id.recognize);
      btnRecognize.setOnClickListener(this);
      break;
      case 'D':
      v = inflater.inflate(R.layout.fragment_tarjeta_caras, container,
                           false);
      wordListView = (ListView) v.findViewById(R.id.word_list);
      mImageView = (ImageView) v.findViewById(R.id.imageViewId);

      mImageView.setImageResource(getResources().getIdentifier(image,
"drawable", context.getPackageName()));
      btnRecognize3 = (ImageButton) v.findViewById(R.id.recognize);
      btnRecognize3.setOnClickListener(this);
      break;
      case 'C':

       v = inflater.inflate(R.layout.fragment_tarjeta_detalle, container,
                           false);
      textoDetalle=(TextView) v.findViewById(R.id.textoDetalle);
      textoDetalle.setText(card.getInstructions());
      wordListView = (ListView) v.findViewById(R.id.word_list_dos);
      btnRecognize2 = (ImageButton) v.findViewById(R.id.recognize2);
      btnRecognize2.setOnClickListener(this);
      break;
       default:
      break;
}

TextView tv = (TextView) v.findViewById(R.id.infoEncuesta);
tv.setText(card.getInfo());
```

Figura 141. Selección automática del layout de tarjeta

6.2.3.4 Reconocedor de voz, síntesis de texto a voz y grabación de resultados

En el método *onCreateView* de la clase ScreenSlidePageFragment, se inicializa además el sintetizador de texto a voz de la forma explicada en el **apartado 6.1.7,** así como el reconocedor de voz, asociando al *ImageButton* que representa el botón del micrófono un escuchador que capture los eventos de tipo pulsación (*onClick*) utilizando el método *setOnClickListener*, como puede apreciarse en la Figura 141. Al realizarse la pulsación del botón con la imagen de un micrófono se ejecuta el método *onClick* para realizar la llamada al *Intent* del reconocedor de voz de Google, cuyo contenido se muestra en la Figura 113 del **Apartado 6.1.6**.

Dentro del método *onActivityResult()*, mostrado también en el **Apartado 6.1.6** y ejecutado como resultado del *Intent* de reconocimiento de voz, se recoge el primer valor de la lista de resultados devueltos. Si se trata del valor "SALIR" significa que el paciente ha pronunciado ese comando especial y se cierra la página de terapia cognitiva invocando al método *exit()* de *ScrenSlidePagerActivity*, como muestra la Figura 142. Se aprecia la forma en la que se puede llamar a un método del *Activity* que alberga el *ViewPager* esperando un periodo determinado para realizar la acción, en este caso 500 milisegundos gracias al uso de la clase *Handler* y su método *postDelayed()*.

Si el resultado es "ESTADÍSTICAS" se invoca de manera análoga el método *stats()* del Activity "padre", donde se creará un Intent para lanzar la actividad que presente la lista de gráficas estadísticas disponibles.

```
if (matches.get(0).toUpperCase().contains("SALIR")) {

	Handler handler = new Handler();
	handler.postDelayed(new Runnable() {
		public void run() {
			((ScreenSlidePagerActivity)getActivity()).exit(getView());}},
500);

}
```

Figura 142. Ejecución de comandos especiales a través de la voz en la terapia cognitiva

Si la respuesta introducida no se trata de un comando especial, se compara con el resultado correcto de la tarjeta, en caso de acierto se lanza un nuevo hilo de ejecución que a los 1500 milisegundos invoque al método *jumpView()* descrito en el **Apartado 6.2.3.6**. Se pausa la ejecución ese tiempo para que pueda finalizar la ejecución de estas dos acciones:

1. Invocar al motor de síntesis de texto a voz para lanzar el mensaje "Respuesta correcta", mediante el método *speakText()*.
2. Insertar el objeto *Record* con el identificador de la tarjeta, el resultado de la operación o Karma (correcta o incorrecta) y la respuesta proporcionada por el paciente. Esta inserción se realiza invocando al método de *ScreenSlidePagerActivity* llamado *insertaRegistroRecord()*, que invoca a la API REST mediante una petición de tipo POST para insertar en base de datos los valores del objeto *Record*.

Por otro lado, si la respuesta proporcionada no era la respuesta correcta de la tarjeta, se incrementa un contador de fallos, si ya se han producido tres fallos consecutivos se invoca a la siguiente tarjeta y se inserta el registro con un karma de valor "N" es decir negativo o respuesta incorrecta. Si es la primera o segunda vez que falla, se realiza la inserción del registro en base de datos pero no se pasa a la siguiente tarjeta, es decir, se le permite que siga interactuando con ella. Para ver de forma gráfica el tratamiento a nivel de código en función de la respuesta proporcionada por el paciente, se muestra el fragmento de código de la aplicación desarrollada en la Figura 143.

```
if (matches.get(0).toUpperCase().contains(kk.toUpperCase())) {


      Handler handler = new Handler();
      handler.postDelayed(new Runnable() {
            public void run() {

      ((ScreenSlidePagerActivity) getActivity()).jumpView(getView());}
            }, 1500);

      r.setKarma("S");
      r.setIdc(card.getId());
      r.setResponse(matches.get(0));
      speakText("Respuesta correcta");
      ((ScreenSlidePagerActivity)getActivity()).insertaRegistroRecord(r);

} else {

      contador_fallos++;

      if (contador_fallos>=3){

      Handler handler = new Handler();
      handler.postDelayed(new Runnable() {
            public void run() {
            ((ScreenSlidePagerActivity) getActivity())
      .jumpView(getView());}}, 1500);
                  speakText("Ha excedido el número máximo de intentos");
             r.setKarma("N");
             r.setIdc(card.getId());
            r.setResponse(matches.get(0));

      ((ScreenSlidePagerActivity)getActivity()).insertaRegistroRecord(r);

      }else{

            speakText("Respuesta incorrecta, "+
card.getInstructions().toLowerCase());
            r.setKarma("N");
            r.setIdc(card.getId());
            r.setResponse(matches.get(0));

         ((ScreenSlidePagerActivity)getActivity()).insertaRegistroRecord(r);

      }
}
```

Figura 143. Código ejecutado en función del éxito de la respuesta del paciente

6.2.3.5 Asociación de adaptador y ViewPager

Una vez desarrollado el contenido de las tarjetas, se procede a crear la vista que contenga el *ViewPager*, donde se realizará la visualización de todas ellas. Se crea el layout tarjeta_deslizante.xml con el contenido de la Figura 144.

```
 <?xml version="1.0" encoding="utf-8"?>
<android.support.v4.view.ViewPager
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/pager"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Figura 144. Layout tarjeta_deslizante.xml

El siguiente paso es crear un adaptador propio, llamado *ScreenSlidePagerAdapter* que hereda de *FragmentStatePagerAdapter*. Este adaptador permite guardar la lista de fragmentos (objetos *ScreenSlidePageFragment*) que se mostrarán en el *ViewPager*. Como muestra la Figura 145, se sobrescriben los métodos *getItem(int position)* para obtener el fragmento alojado en la posición indicada, *getCount()* para obtener el número total de fragmentos y *setItem(ScreenSlidePageFragment fr, int position)* para añadir un nuevo fragmento a la lista.

```
private class ScreenSlidePagerAdapter extends FragmentStatePagerAdapter {

private List<ScreenSlidePageFragment> fragments = new
ArrayList<ScreenSlidePageFragment>();

public ScreenSlidePagerAdapter(
      android.support.v4.app.FragmentManager fragmentManager,
            List<ScreenSlidePageFragment> fragments) {
           super(fragmentManager);
           this.fragments = fragments;
     }

     @Override
     public android.support.v4.app.Fragment getItem(int position) {
            return fragments.get(position);
     }

     @Override
     public int getCount() {
           return this.fragments.size();
     }

     public void setItem(ScreenSlidePageFragment fr, int position) {
           fragments.add(fr);
     }
}
```

Figura 145. Adaptador personalizado para ViewPager

Para finalizar, es necesario añadir las tarjetas (*fragments*) que va a mostrar el *ViewPager* y conectar el adaptador, *ScreenSlidePagerActivity* es la *Activity* creada para tal efecto. El código necesario aparece en la Figura 146, cuyas acciones se resumen en:

- En primer lugar, es necesario hacer que esta actividad extienda de *FragmentActivity*.
- Después se establece como layout de la pantalla el que contiene el *ViewPager* (tarjeta_deslizante.xml), se recupera la instancia de ese elemento.
- Se inicializa el adaptador creado anteriormente *ScreenSlidePagerAdapter* pasando como parámetro la lista que contiene los fragmentos que simbolizan cada una de las tarjetas que deberá realizar el usuario en el nivel de dificultad actual.
- Los usuarios no deben poder pasar de una tarjeta a otra realizando el gesto de deslizar el dedo por la pantalla, acción denominada ***swipe***, por lo que se deshabilita esta opción añadiendo un evento de tipo OnTouchListener a la instancia del *ViewPager*, de manera que cuando suceda el evento y se invoque el método *onTouch()* únicamente se devuelva el valor *true*.
- Se establece el adaptador creado al *ViewPager*, en ese momento se mostrará en pantalla la tarjeta de la primera posición de la lista.

```
setContentView(R.layout.tarjeta_deslizante);

// Instantiate a ViewPager and a PagerAdapter.
mPager = (ViewPager) findViewById(R.id.pager);

mPagerAdapter = new ScreenSlidePagerAdapter(
                                  getSupportFragmentManager(), fragments);

// Para deshabilitar el swipe manual
mPager.setOnTouchListener(new OnTouchListener() {
      @Override
      public boolean onTouch(View v, MotionEvent event) {
            return true;
      }
});

mPager.setAdapter(mPagerAdapter);
```

Figura 146. Asociación de ViewPager con adaptador desarrollado

6.2.3.6 Transición entre tarjetas

Para que la aplicación realice la transición automática entre las tarjetas, se recurre a la invocación del método *jumpView()* alojado en la actividad *ScreenSlidePagerActivity*. Cuando el paciente contesta correctamente a una tarjeta, o se ha superado el límite de tres respuestas incorrectas, se invoca a este método desde el *ScreenSlidePageFragment* que representa esa tarjeta.

Una vez dentro del método, se recupera el número de tarjeta que se está mostrando en el *ViewPager*, instanciado aquí como *mPager*. Si el número coincide con los elementos disponibles en el vector de tarjetas, significa que es el último elemento, por lo que se procede a invocar al método *uploadToServerUpdateUser()* que actualizará el nivel de dificultad de las tarjetas al inmediatamente posterior. Además se invocará mediante *Intent* a la *Activity Level* que informa del fin del nivel de dificultad, cuyo diseño se especifica en el **Apartado 5.6.6**.

Si la tarjeta actual no es la última del vector, se aumenta la posición de la página mostrada en el ViewPager mediante el método *setCurrentItem()*.

El código del método *jumpView* se proporciona en la Figura 147.

```
public void jumpView(View view) {

	//Si estamos en la última tarjeta a mostrar, realizamos la acción correspondiente
	if(this.mPager.getCurrentItem()==prueba.size()-1){
		//TODO de momento salimos al menú principal, pero lo suyo sería mostrar un mensaje informativo
		//ScreenSlidePagerActivity.this.finish();
		Intent i = new Intent(ScreenSlidePagerActivity.this, Level.class);
		i.putExtra("user", user);
		i.putExtra("surname", surname);
		// paso el identificador de usuario la actividad principal
		i.putExtra("idu", idu);
		i.putExtra("level", level);
		i.putExtra("language", language);
		uploadToServerUpdateUser();
		startActivity(i);
		ScreenSlidePagerActivity.this.finish();
	}else{
	this.mPager.setCurrentItem(this.mPager.getCurrentItem() + 1);
	}
}
```

Figura 147. Gestión de la transición entre tarjetas

6.2.4 Módulo estadístico

Este módulo se encarga de presentar las gráficas estadísticas con información referente al progreso de la terapia cognitiva del paciente. Su implementación se reutiliza en las dos aplicaciones desarrolladas: HealthCoach y HealthCoach Admin, cuya interfaz gráfica puede observarse en el **Apartado B.1.6** perteneciente al **Anexo B**. De esa manera, tanto el paciente como su responsable pueden ver la evolución a lo largo de los distintos niveles de dificultad.

La primera parte del módulo se implementa en la *Activity* llamada *StatisticsLauncher* con el propósito de crear un **menú** que liste los tipos de gráficas estadísticas disponibles:

- **Aciertos por ámbito cognitivo.**
- **Errores por ámbito cognitivo.**
- **Racha de aciertos por nivel.**

El desarrollo del menú sigue la misma filosofía que el otro menú implementado en el **Apartado 6.2.2**, pero enlazando con los tres Activity encargados de gestionar cada una de las estadísticas que acaban de exponerse.

Estos tres Activity hacen uso de la API **AndroidPlot** [71] presentada en el **Apartado 3.1.11**. Para utilizarla, es necesario descargar la última versión de la librería disponible en la url: http://androidplot.com/download/ (*androidplot-core-0.6.1.jar* en el momento de implementar este

proyecto) e importarla al proyecto de Eclipse mediante clic derecho en la carpeta padre del proyecto, navegando hasta *Properties>Java Build Path>Libraries* y seleccionando la opción *Add External JARs* para elegir la librería de extensión .jar descargada. A partir de ese momento es posible importar desde cualquier Activity del proyecto las clases proporcionadas por AndroidPlot.

A continuación, se describe la implementación desarrollada en cada una de las clases Activity lanzadas desde el menú:

BarPlotSerieActivity

Esta clase es la encargada de generar la gráfica estadística relativa a la racha máxima de aciertos del paciente en cada uno de los niveles de dificultad. Su interfaz puede consultarse en el **Apartado B.1.6.3** del **Anexo B**.

Como es habitual en casi todos los Activity, la primera acción realizada en el método *onCreate()* consiste en recuperar los datos relativos al usuario (identificador, nombre, correo electrónico, etc.) para, a continuación, invocar la API REST alojada en el servidor remoto mediante una petición de tipo GET a la URL mostrada en la Figura 148. Esto permite obtener de base de datos la información de las respuestas proporcionadas por el paciente y almacenadas en la tabla RECORS.

```
private static String urlUserRecords = "http://sindecito.noip.me/api/records/";

String jsonStr = sh.makeServiceCall(urlUserRecords + idu,
                    ServiceHandler.GET);
```

Figura 148. Petición GET a API remota para obtener respuestas de usuario

Ya en el código PHP la petición se filtra hacia el método *getRecordsByUser($id)* que ejecuta la sentencia SQL mostrada en la Figura 149 para obtener de forma cronológica el acierto/error y dificultad de la tarjeta asociada a cada una de las respuestas proporcionadas.

```
function getRecordsByUser($id) {
    $sql = "SELECT R.KARMA, C.DIFFICULTY FROM RECORDS R, CARDS C WHERE
R.IDC=C.IDC AND R.IDU=:id ORDER BY R.IDR ASC";
    try {
        $dbCon = getConnection();
        $stmt = $dbCon->prepare($sql);
        $stmt->bindParam("id", $id);
        $stmt->execute();
        $records = $stmt->fetchAll(PDO::FETCH_OBJ);
        $dbCon = null;
        echo '{"records": ' . json_encode($records) . '}';
    } catch(PDOException $e) {
        echo '{"error":{"text":'. $e->getMessage() .'}}';
    }
}
```

Figura 149. Función PHP de consulta de respuestas de usuario

Una vez recuperada la información del objeto JSON que contiene la respuesta, se filtra el índice de dificultad y en caso de respuesta afirmativa se incrementa el contador asociado a ese nivel, en caso contrario se restablece a cero. Una vez procesado por completo el objeto JSON se dispone de los seis contadores de nivel con el valor de la mayor racha conseguida.

La gráfica se representa dentro de un LinearLayout gracias al componente com.androidplot.xy.XYPLOT, con la configuración mostrada a continuación. En él se definen todos los tamaños y etiquetas referentes a los ejes de abscisas y ordenadas. El contenido del componente se muestra en la Figura 150.

```
<com.androidplot.xy.XYPlot
        android:id="@+id/mySimpleXYPlot"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1"
        androidPlot.domainLabel="Nivel tarjetas"

androidPlot.domainLabelWidget.labelPaint.textSize="@dimen/domain_label_font_size"
        androidPlot.graphWidget.domainGridLinePaint.alpha="0"

androidPlot.graphWidget.domainLabelPaint.textSize="@dimen/domain_tick_label_font_
size"

androidPlot.graphWidget.domainOriginLabelPaint.textSize="@dimen/domain_tick_label
_font_size"
        androidPlot.graphWidget.domainOriginLinePaint.alpha="0"
        androidPlot.graphWidget.gridBackgroundPaint.color="#FFFFFF"
        androidPlot.graphWidget.gridLinePaint.color="#000000"
        androidPlot.graphWidget.marginBottom="25dp"
        androidPlot.graphWidget.marginLeft="15dp"
        androidPlot.graphWidget.marginRight="10dp"
        androidPlot.graphWidget.marginTop="20dp"

androidPlot.graphWidget.rangeLabelPaint.textSize="@dimen/range_tick_label_font_si
ze"

androidPlot.graphWidget.rangeOriginLabelPaint.textSize="@dimen/range_tick_label_f
ont_size"
        androidPlot.legendWidget.heightMetric.value="25dp"
        androidPlot.legendWidget.iconSizeMetrics.heightMetric.value="15dp"
        androidPlot.legendWidget.iconSizeMetrics.widthMetric.value="15dp"
        androidPlot.legendWidget.positionMetrics.anchor="right_bottom"

androidPlot.legendWidget.textPaint.textSize="@dimen/legend_text_font_size"
        androidPlot.rangeLabel="Número de aciertos"

androidPlot.rangeLabelWidget.labelPaint.textSize="@dimen/range_label_font_size"
        androidPlot.title="Racha máxima de aciertos"
        androidPlot.titleWidget.labelPaint.textSize="@dimen/title_font_size" />
```

Figura 150. Componente XYPlot para representación de gráfica estadística

Desde la clase *BarPlotSerieActivity* se inicializa el objeto XYPlot que referencia al componente gráfico recién descrito y así configurar el resto de propiedades gráficas. Entre los métodos utilizados destaca *SimpleXYSeries()* para establecer los valores del eje X y eje Y, relativos al número de nivel y a la racha máxima de aciertos respectivamente:

```
series1 = new SimpleXYSeries(Arrays.asList(xValues),
                          Arrays.asList(seriesUserNumbers), user);
```

Además se implementan eventos de tipo *OnTouchListener* para captar las pulsaciones sobre las barras y presentar información adicional como por ejemplo: “Racha de 2 aciertos en el nivel 4 para David”.

BarPlotActivity

Esta clase es la encargada de generar la gráfica que representa el número de errores por ámbito cognitivo cometidos por el paciente a lo largo de la terapia. Su interfaz puede consultarse en el **Apartado B.1.6.2** del **Anexo B**. La implementación es similar a *BarPlotSerieActivity* pero en esta ocasión se necesita una doble llamada a la API REST, para obtener tanto el número total de tarjetas por ámbito cognitivo (mediante la URL definida en *urlTotalStats*), como el número de tarjetas de cada ámbito contestadas correctamente (mediante la URL *urlUserStats*), como muestra la Figura 151.

```
// URL to get total stats JSON
private static String urlTotalStats =
"http://192.168.1.19:8080/api/totalCognitiveStats";

// Making a request to url and getting response
      String jsonStr = sh.makeServiceCall(urlTotalStats, ServiceHandler.GET);

// URL to get user stats JSON
private static String urlUserStats = "http://
192.168.1.19:8080/api/cognitiveStats/";


// Making a request to url and getting response
 String jsonStr = sh.makeServiceCall(urlUserStats+idu, ServiceHandler.GET);


function getCognitiveStats($id) {
    $sql = "SELECT `COGNITIVE`, COUNT(*) FROM CARDS WHERE IDC IN ( SELECT IDC
FROM RECORDS WHERE IDU=:id AND KARMA=:kar) GROUP BY `COGNITIVE`";
.
.
}

function getTotalCognitiveStatsBis() {
    $sql = "SELECT `COGNITIVE`, COUNT(*) FROM CARDS GROUP BY `COGNITIVE`";
.
.
}
```

Figura 151. Obtención de estadísticas mediante invocación a API remota

Con estos datos puede generarse una gráfica de tipo XYPlot mediante la inclusión de dos series (clase *XYSerie*), que comparten eje de abscisas y ordenadas para el ámbito cognitivo y número de aciertos respectivamente. Al ver de forma superpuesta las barras, de distinto color, se aprecia una posible diferencia entre el total de tarjetas de ese ámbito con el número de aciertos, lo que nos da de manera indirecta la tasa de error.

SimplePiecharActivity

Esta última actividad genera la gráfica de tipo "tarta" que representa los aciertos totales, con indiferencia del nivel de la tarjeta, que se han producido en cada uno de los ámbitos cognitivos. Su interfaz puede consultarse en el **Apartado B.1.6.1** del **Anexo**.

La consulta a la API REST es la misma que la realizada en *BarPlotActivity*, sin embargo la representación gráfica en forma de tarta requiere utilizar una nueva clase de AndroidPlot denominada *PieChart*, como se observa en la porción de código mostrada en la Figura 152. Para definir cada "porción" se utiliza la clase *Segment*, inicializándola con el nombre de la porción y su valor. Una vez creado el *Segment* es necesario configurarlo para establecerle el color, tamaño de la fuente, etc. Para ese propósito se recurre a una nueva clase, *SegmentFormatter*, a la que se le pasan estas propiedades en formato *.xml*. Para finalizar se añade al objeto PieChart la serie mediante el método *addSeries(Segment, SegmentFormatter).*

```
pie = (PieChart) findViewById(R.id.mySimplePieChart);

s1 = new Segment("Memoria", memoria);
s2 = new Segment("Gnosias", gnosias);
s3 = new Segment("Orientación", orientacion);
s4 = new Segment("Aritmética", aritmetica);
s5 = new Segment("Comunicación", comunicacion);
s6 = new Segment("Ejecutiva", ejecutiva);

SegmentFormatter sf1 = new SegmentFormatter();
sf1.configure(getApplicationContext(), R.xml.pie_segment_formatter1);
pie.addSeries(s1, sf1);
.
.
.
```

Figura 152. Definición de segmentos y series de objeto PieChart

AndroidPlot pone a disposición de los desarrolladores documentación y código de ejemplo en la siguiente ruta: http://androidplot.com/docs/.

6.2.5 Módulo de estadio de Alzheimer

Uno de los principales objetivos de este proyecto consiste en, con la realización de los ejercicios cognitivos, llegar a generar una estimación del grado de Alzheimer del paciente.

Basándose en el diseño funcional del **Apartado 5.9**, es necesaria la implementación de este módulo en las aplicaciones HealthCoach y HealthCoach Admin.

6.2.5.1 Cálculo desde HealthCoach

Dentro de HealthCoach, el proceso de cálculo se inicia una vez finalizado el proceso de adaptabilidad descrito en el **Apartado 6.2.3.1** si el Vector de tarjetas resultantes está vacío, es decir, se ha llegado a un punto en el que el sistema no tiene mas tarjetas para presentar al usuario. Este proceso se ejecuta de manera transparente al usuario.

Las respuestas obtenidas a cada una de las tarjetas implicadas en el cálculo del grado de Alzheimer se obtienen mediante una llamada asíncrona con la clase AsyncTask a la API REST, cuyo código PHP acaba realizando una consulta a la tabla RECORDS para obtener el valor del campo KARMA, que alberga el valor "S" para respuestas correctas o "N" en caso contrario. Una vez recuperado el objeto JSON se obtiene el resultado de las tarjetas, si un valor necesario para el cálculo no existe porque no se ha contestado (debido al proceso de adaptabilidad) se marca como respuesta incorrecta.

Una vez obtenida la estimación del grado de enfermedad, comprendida entre los niveles 1 y 7, se envía el resultado a su responsable a través de correo electrónico gracias al uso de Javamail, introduciendo en el cuerpo del mensaje código HTML, como muestra la Figura 153. El correo resultante puede consultarse en el **Apartado B.2.8**.

```
sendMail("correo@gmail.com", "HealthCoach - Estimación estadio usuario "+user+"
"+surname,"<html><body><h1 align=\"center\">Resultado estimación</h1><p
align=\"center\">El estadio estimado de la enfermedad para el paciente
<b>"+user+" "+surname+"</b> es</p><p align=\"center\"><font
size=\"12\"><b>"+estadio+"</b></font></p><p align=\"center\">Acceda a la
aplicación de administración de pacientes para obtener más
información.</p></body></html></body></html>");
```

Figura 153. Generación de correo electrónico con el estadio de Alzheimer del paciente

6.2.5.2 Cálculo desde HealthCoach Admin

En el caso de la aplicación de responsables, el cálculo del estadio de la enfermedad del paciente seleccionado se ejecuta de forma explícita gracias a la opción de menú existente, como puede comprobarse en el **Apartado B.2.5** del **Anexo B**. La Activity invocada, denominada *Degree* se encarga de invocar al mismo servicio de la *API REST* que devuelve el objeto *JSON* con la información referente a las respuestas proporcionadas por el paciente en las tarjetas necesarias para realizar el cálculo del grado de Alzheimer.

Como puede observarse en el Apartado **B.2.7**, se utiliza un TextView para mostrar el resultado y en la parte inferior se implementa otro menú de la forma habitual, utilizando elementos *TextView* con setOnClickListener() definidos para capturar la pulsación y lanzar el código necesario para realizar la siguiente acción, en este caso para volver a la pantalla de Menú principal lanzando el Intent que invoque a la *Activity HiScreen* o resetear los datos del paciente referentes al avance de la terapia cognitiva (explicado en el siguiente apartado).

6.2.5.3 Resetear datos de paciente

Después de consultar desde la aplicación HealthCoach Admin el estadio de Alzheimer se presenta al usuario la opción de resetear los datos relativos a la evolución del paciente en la terapia cognitiva.

Al pulsar la opción de menú, el listener asociado lanza un *AlertDialog*, cuya implementación se muestra en la Figura 154, para pedir confirmación al usuario sobre la opción que va a realizar.

```
AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);
alertDialogBuilder.setTitle("Confirmación reseteo");
alertDialogBuilder.setMessage("¿Desea resetear el progreso del usuario? Si
confirma, esta información no será
recuperable.").setCancelable(false).setPositiveButton("Sí", new
DialogInterface.OnClickListener() {
       public void onClick(DialogInterface dialog, int id) {
              dialog.cancel();
              estadio=-1;

              // Pone a 1 el nivel del usuario, para visualizar tarjetas de ese
nivel
             uploadToServerUpdateUser();

             // Establece a -1 el estadio del usuario
             uploadToServerDegreeUser();

              // Borra todos los registros de interacción con las tarjetas
              uploadToServerDeleteRecord();

             //Después de resetear vamos a la pantalla de selección de usuario
              seleccionUsuario();

              }
});
alertDialogBuilder.setNegativeButton("No",
      new DialogInterface.OnClickListener() {
              public void onClick(DialogInterface dialog, int which) {

              }
            });
AlertDialog alertDialog = alertDialogBuilder.create();
alertDialog.show();
```

Figura 154. Confirmación de reseteo de datos de usuario mediante AlertDialog

Si pulsa en “No” se destruye el AlertDialog, si por el contrario se elige “Si” se realiza una triple invocación a la API REST:

1. Se realiza una petición de tipo UPDATE al servicio REST remoto para establecer el nivel actual del usuario a 1en la tabla USERS, con esto logramos que vuelvan a presentarse
2. Petición HTTP de tipo UPDATE para actualizar el estadio del usuario en la tabla USERS a -1.
3. Petición HTTP de tipo DELETE para borrar todos los registros de interacción con las tarjetas por parte del usuario seleccionado. Por tanto, invoca una sentencia SQL para borrar todas las entradas de la tabla RECORDS con el identificador de usuario (IDU) informado por parámetro. La Figura X muestra la invocación realizada desde la aplicación Android.

```
HttpClient httpclient = new DefaultHttpClient(params);

HttpDelete httpDelete = new
HttpDelete("http://192.168.1.19:8080/api/records/"+idu);

HttpResponse httpResponse = httpclient.execute(httpDelete);
```

Figura 155. Invocación a API remota mediante petición de tipo DELETE

Capítulo 7

Evaluación

Para evaluar el sistema HealhCoach, se han diseñado dos cuestionarios que recogen las valoraciones subjetivas de los usuarios que interactúan con las dos aplicaciones Android desarrolladas en este proyecto.

Los cuestionarios han sido generados mediante la herramienta web ***Google Forms*** [87], proporcionada por Google de forma gratuita dentro de su *suite* ofimática en la nube, denominada *Google Docs* [88]. Una de sus ventajas principales es la posibilidad de generar las encuestas en la web y distribuir los enlaces de acceso público a los futuros usuarios encuestados.

Como estructura general, se han diseñado preguntas con cinco posibles respuestas para ambos cuestionarios, entre las cuáles, el usuario debe elegir la opción que más se ajuste a la realidad percibida.

7.1 Evaluación de HealthCoach

Esta encuesta se diseña con el objetivo de obtener la opinión de los **pacientes** y **responsables** en cuanto a facilidad de interacción con la aplicación, ocupando un papel importante la interacción táctil pero siendo especialmente crítica la experiencia de uso mediante la voz durante la realización de las tarjetas cognitivas.

Debido a la imposibilidad de probar la aplicación **HealthCoach** con enfermos reales de Alzheimer**,** se ha decidido realizar una evaluación a través de tres personas con escasa experiencia con tecnologías de la información, con edades comprendidas entre los 58 y 67 años.

La Figura 156 muestra el cuestionario de evaluación proporcionado tanto al grupo que representa los pacientes como al de responsables, a través de la URL http://goo.gl/forms/r9DLKvXPLe. Consta de doce preguntas, realizadas con los siguientes propósitos:

- **Preguntas 1 y 2**. Se centran en la experiencia de uso mediante interacción táctil con la aplicación.

- **Preguntas 3, 4, 5 y 6**. Evalúan de uso mediante interacción oral en las tarjetas de terapia cognitiva.

- **Pregunta 7**. Mide la sensación de adaptabilidad en las tarjetas de ejercicios cognitivos presentados al usuario.

- **Pregunta 8.** Valora la sensación de mejora de la memoria por parte del usuario.

- **Preguntas 9 y 10.** Miden el grado de dificultad percibido para completar los procesos de registro e identificación.

- **Pregunta 11.** Evalúa la dificultad del usuario para interpretar las gráficas estadísticas relativas al progreso de la terapia cognitiva.

- **Pregunta 12.** Mide de forma global el grado de satisfacción del usuario con el sistema HealthCoach.

Encuesta de aplicación HealthCoach

*Obligatorio

1 - Indique la dificultad de manejo táctil de la aplicación. *

- Muy fácil.
- Fácil.
- Normal.
- Difícil.
- Muy difícil.

2 - ¿Le ha resultado sencillo decidir qué hacer en cada momento para conseguir lo que deseaba dentro de la aplicación? *

- No, ha resultado imposible.
- No, he encontrado gran dificultad.
- Sí, aunque con cierta dificultad
- Sí, ha resultado sencillo.
- Sí, ha resultado muy sencillo.

3 - ¿Qué tal ha comprendido la aplicación sus respuestas a los ejercicios? *

- Muy mal.
- Mal.
- Regular.
- Bien.
- Muy bien.

4 - ¿Ha comprendido los mensajes de voz generados por la aplicación? *

- Muy mal.
- Mal.
- Regular.
- Bien.
- Muy bien.

5 - ¿Se han producido errores en la interacción de voz durante la realización de los ejercicios? *

- No se han producido.
- Sí, muchos. Impidiendo la realización de los ejercicios.
- Sí, bastantes. Provocando gran dificultad en el desarrollo de los ejercicios.
- Sí, algunos. Dificultando la realización de los ejercicios.
- Sí, algunos. Pero no han afectado en el desarrollo de los ejercicios.

6 - Indique de forma general la dificultad de interacción oral con las tarjetas de ejercicios. *

- ○ Muy difícil.
- ○ Difícil.
- ○ Normal
- ○ Fácil.
- ○ Muy fácil.

7 - ¿En qué grado ha percibido una adaptación en el nivel de dificultad de las tarjetas según iba progresando en la realización de los ejercicios? *

- ○ Nada.
- ○ Poco.
- ○ Algo.
- ○ Bastante.
- ○ Mucho.

8 - ¿Considera que la realización de los ejercicios le ha ayudado a ejercitar la memoria? *

- ○ Nada.
- ○ Poco.
- ○ Algo.
- ○ Bastante.
- ○ Mucho.

9 - ¿Considera intuitivo el proceso de registro en la aplicación? *

- ○ Nada intuitivo.
- ○ Algo intuitivo.
- ○ Intuitivo.
- ○ Bastante intuitivo.
- ○ Muy intuitivo.

10 - ¿Considera intuitivo el proceso de identificación en la aplicación? *

- ○ Nada intuitivo.
- ○ Algo intuitivo
- ○ Intuitivo.
- ○ Bastante intuitivo.
- ○ Muy intuitivo.

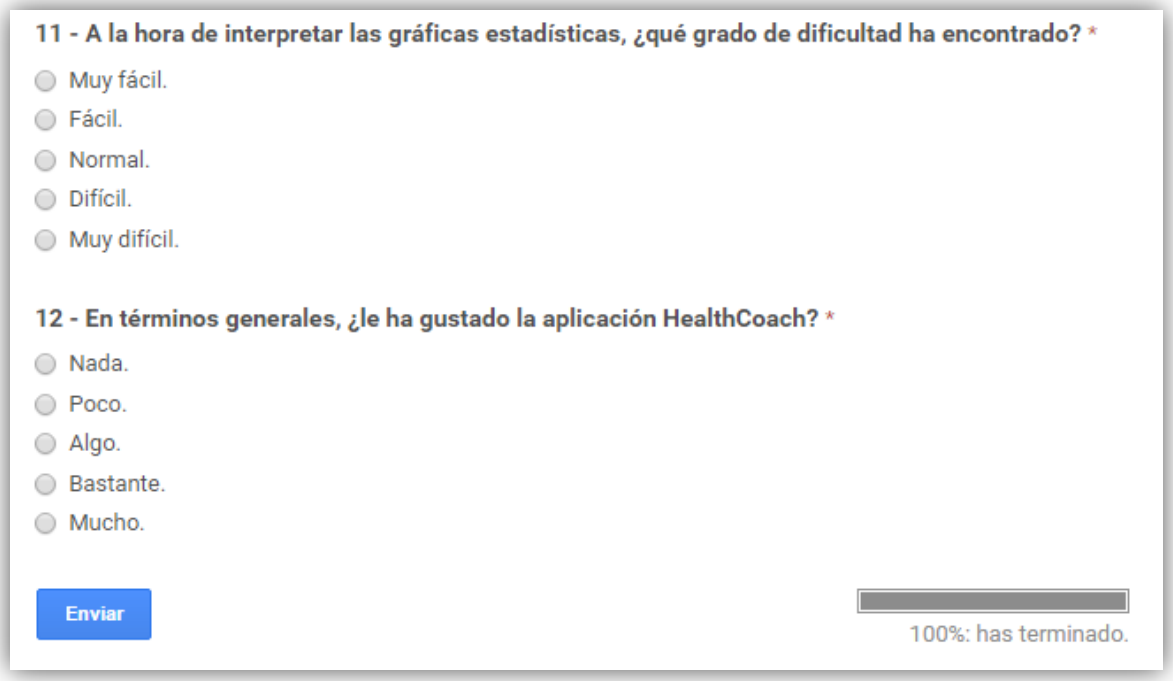
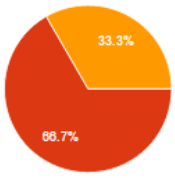


Figura 156. Encuesta de evaluación para aplicación de pacientes

7.1.1 Resultados de la evaluación de HealthCoach

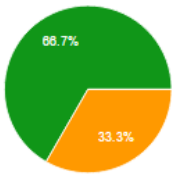
Una vez presentada la aplicación a los pacientes descritos en el apartado anterior (**Apartado 7.1**), mediante unas mínimas explicaciones sobre el funcionamiento de la misma, y dejar que interactuaran con ella siempre de forma guiada, se les instó a completar el cuestionario de evaluación. Los resultados obtenidos se recuperan del servicio *Google Forms*, cuyas estadísticas pueden observarse en la Figura 157.

1 - Indique la dificultad de manejo táctil de la aplicación.



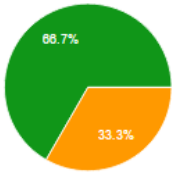
Muy fácil.	0	0%
Fácil.	2	66.7%
Normal.	1	33.3%
Difícil.	0	0%
Muy difícil.	0	0%

2 - ¿Le ha resultado sencillo decidir qué hacer en cada momento para conseguir lo que deseaba dentro de la aplicación?



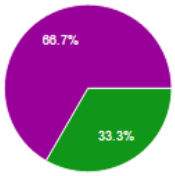
No, ha resultado imposible.	0	0%
No, he encontrado gran dificultad.	0	0%
Sí, aunque con cierta dificultad	1	33.3%
Sí, ha resultado sencillo.	2	66.7%
Sí, ha resultado muy sencillo.	0	0%

3 - ¿Qué tal ha comprendido la aplicación sus respuestas a los ejercicios?



Muy mal.	0	0%
Mal.	0	0%
Regular.	1	33.3%
Bien.	2	66.7%
Muy bien.	0	0%

4 - ¿Ha comprendido los mensajes de voz generados por la aplicación?

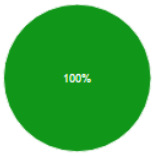


Muy mal.	0	0%
Mal.	0	0%
Regular.	0	0%
Bien.	1	33.3%
Muy bien.	2	66.7%

5 - ¿Se han producido errores en la interacción de voz durante la realización de los ejercicios?

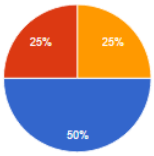
No se han producido.	0	0%
Sí, muchos. Impidiendo la realización de los ejercicios.	0	0%
Sí, bastantes. Provocando gran dificultad en el desarrollo de los ejercicios.	0	0%
Sí, algunos. Dificultando la realización de los ejercicios.	0	0%
Sí, algunos. Pero no han afectado en el desarrollo de los ejercicios.	3	100%

6 - Indique de forma general la dificultad de interacción oral con las tarjetas de ejercicios.



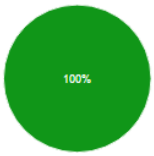
Muy difícil.	0	0%
Difícil.	0	0%
Normal	0	0%
Fácil.	4	100%
Muy fácil.	0	0%

7 - ¿En qué grado ha percibido una adaptación en el nivel de dificultad de las tarjetas según iba progresando en la realización de los ejercicios?



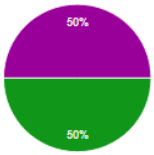
Nada.	2	50%
Poco.	1	25%
Algo.	1	25%
Bastante.	0	0%
Mucho.	0	0%

8 - ¿Considera que la realización de los ejercicios le ha ayudado a ejercitar la memoria?



Nada.	0	0%
Poco.	0	0%
Algo.	0	0%
Bastante.	4	100%
Mucho.	0	0%

9 - ¿Considera intuitivo el proceso de registro en la aplicación?



Nada intuitivo.	0	0%
Algo intuitivo.	0	0%
Intuitivo.	0	0%
Bastante intuitivo.	2	50%
Muy intuitivo.	2	50%

10 - ¿Considera intuitivo el proceso de identificación en la aplicación?

Nada intuitivo.	0	0%
Algo intuitivo	0	0%
Intuitivo.	0	0%
Bastante intuitivo.	0	0%
Muy intuitivo.	4	100%

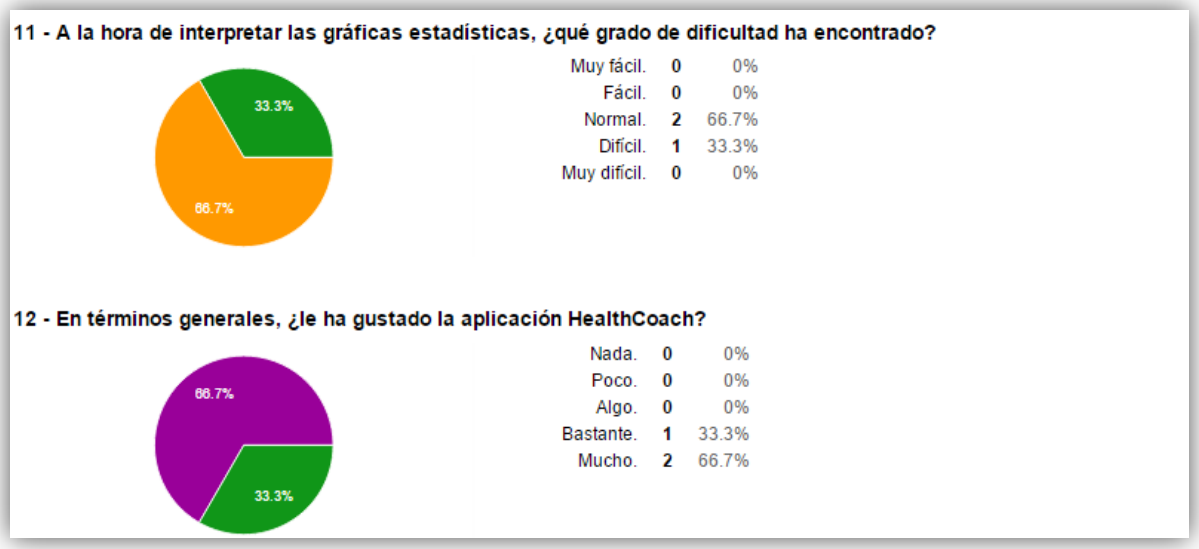


Figura 157. Resultados de la evaluación de la aplicación HealthCoach

A partir de las estadísticas generadas mediante las respuestas de los usuarios encuestados, se obtienen estas conclusiones:

- De forma general, la interacción táctil con la aplicación se realiza de manera satisfactoria, predominando la percepción de un buen nivel de usabilidad.

- En cuanto a la interacción oral con las tarjetas cognitivas, se percibe un buen funcionamiento del reconocedor de voz, a pesar de producirse errores puntuales de interpretación en la respuesta proporcionada por el paciente. Se deben a ruidos ambientales o a pronunciaciones poco claras. Los mensajes de voz generados por el sistema son captados de forma correcta por los usuarios. De forma general, consideran fácil la interacción oral con la aplicación.

- El proceso de adaptabilidad pasa prácticamente desapercibido para los usuarios, como era de esperar, al ser un sistema transparente para el paciente.

- Existe unanimidad en la creencia de los usuarios en relación al entrenamiento de la memoria gracias al uso de la aplicación HealthCoach, considerando que les ha ayudado bastante a ejercitarla.

- Los procesos de registro e identificación de usuario se realizan sin ningún inconveniente, considerándolos muy intuitivos.

- La interpretación de las gráficas estadísticas generadas a partir del progreso en la terapia presenta cierto nivel de dificultad para los pacientes.

- Para finalizar, se observa cómo la aplicación HealthCoach ha gustado a los pacientes que la han utilizado.

7.2 Evaluación de HealthCoach Admin

El segundo cuestionario de evaluación pretende obtener las opiniones de los **responsables**, quienes monitorizan la evolución de sus pacientes en la terapia cognitiva utilizando la aplicación HealthCoach Admin. Al igual que en la evaluación anterior (**Apartado 7.1**), es importante conocer la valoración de estos usuarios en materia de interacción con la aplicación, pero es aún más importante la información referente al grado de satisfacción con la monitorización de la terapia y proceso de cálculo de grado de la enfermedad del paciente.

Esta evaluación la han realizado, después de probar la aplicación, dos trabajadoras del ámbito sanitario con una dilatada experiencia en el cuidado de enfermos con síntomas de demencia. El cuestionario que se les ha facilitado (Figura 158) se encuentra disponible en la URL: http://goo.gl/forms/nUsWzNn8HF.

Consta de trece preguntas que se centran en los aspectos siguientes:

- **Preguntas 1 y 2**. Valoran la experiencia de uso mediante interacción táctil con la aplicación.
- **Preguntas 3, 4**. Miden el grado de dificultad percibido para completar los procesos de registro y selección de paciente.
- **Pregunta 5**. Evalúa la capacidad de la aplicación para llevar un seguimiento del progreso de la terapia cognitiva del paciente.
- **Preguntas 6 y 7.** Evalúan la dificultad percibida para obtener información de las gráficas estadísticas disponibles así como su utilidad práctica.
- **Preguntas 8, 9 y 10.** Miden el grado de satisfacción con la información proporcionada en los correos electrónicos enviados al responsable, tanto la relativa al progreso de la terapia (resultados por niveles, adaptabilidad aplicada), como la referente al resultado de la estimación del grado de Alzheimer del paciente. Por último, requieren opinión del encuestado referente a la necesidad de ampliar la información proporcionada en dichos correos.
- **Preguntas 11.** Comprueba si el resultado del nivel estimado de Alzheimer para el paciente se asemeja al diagnosticado por medios clínicos o al percibido de forma subjetiva por el responsable.
- **Pregunta 12.** Evalúa si la información proporcionada en relación a la terapia cognitiva del paciente es suficiente o si por el contrario sería necesario ampliarla.
- **Pregunta 13.** Mide de forma global el grado de satisfacción del usuario con el sistema HealthCoach.

Encuesta de aplicación HealthCoach Admin

*Obligatorio

1 - Indique la dificultad de manejo táctil de la aplicación *

- Muy fácil.
- Fácil.
- Normal.
- Difícil.
- Muy difícil.

2 - ¿Le ha resultado sencillo decidir qué hacer en cada momento para conseguir lo que deseaba dentro de la aplicación? *

- No, ha resultado imposible.
- No, he encontrado gran dificultad.
- Sí, aunque con cierta dificultad.
- Sí, ha resultado sencillo.
- Sí, ha resultado muy sencillo.

3 - ¿Considera intuitivo el proceso de selección de paciente? *

- Nada intuitivo.
- Algo intuitivo.
- Intuitivo.
- Bastante intuitivo.
- Muy intuitivo.

4 - ¿Considera intuitivo el proceso de registro en la aplicación? *

- Nada intuitivo.
- Algo intuitivo.
- Intuitivo.
- Bastante intuitivo.
- Muy intuitivo.

5 - ¿Puede realizarse de forma adecuada el seguimiento del paciente? *

- No, nada adecuada.
- Poco adecuada.
- Adecuada.
- Bastante adecuada.
- Muy adecuada.

6 - A la hora de interpretar las gráficas estadísticas, ¿qué grado de dificultad ha encontrado? *

- Muy fácil.
- Fácil.
- Normal.
- Difícil.
- Muy difícil.

7 - ¿Considera de utilidad la información proporcionada por las gráficas estadísticas? *

- Nada útil.
- Poco útil.
- Útil.
- Bastante útil.
- Muy útil.

8 - ¿Considera apropiado el envío de correos electrónicos con la información del progreso del paciente en la terapia cognnitiva? *

- No, nada apropiado.
- Poco apropiado.
- Apropiado.
- Algo apropiado.
- Muy apropiado.

9 - ¿Considera apropiado el envío de correos electrónicos con el cálculo estimado del grado de Alzheimer del paciente? *

- No, nada apropiado.
- Poco apropiado.
- Apropiado.
- Algo apropiado.
- Muy apropiado.

10 - ¿Considera necesario ampliar la información contenida en los correos electrónicos que reportan el progreso de la terapia cognitiva del paciente? *

- No, nada necesario.
- Algo necesario.
- Necesario.
- Bastante necesario.
- Muy necesario.

11 - ¿El nivel estimado de la enfermedad coincide con el percibido de forma subjetiva? *

- Nada.
- Poco.
- Se acerca.
- Coincide bastante.
- Coincide plenamente.

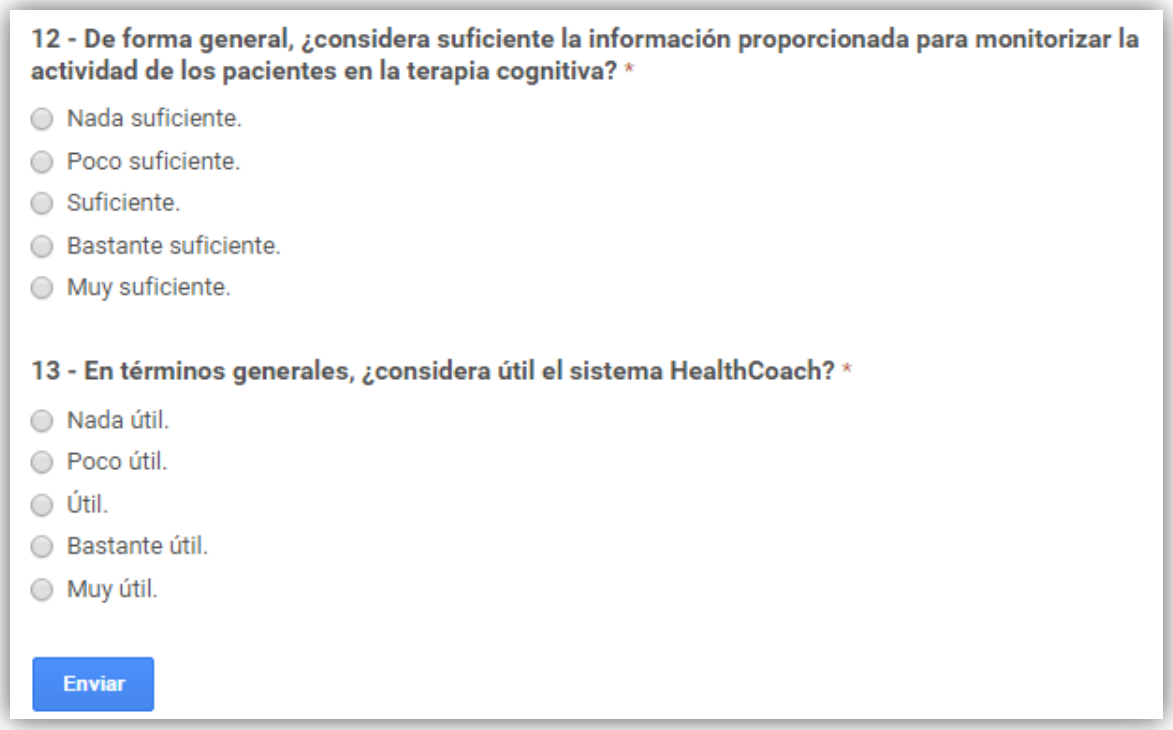
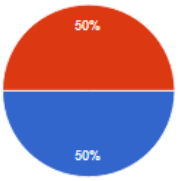


Figura 158. Encuesta de evaluación para aplicación de responsables

7.2.1 Resultados de la evaluación de HealthCoach Admin

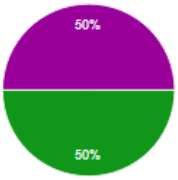
Después de presentar la aplicación a las dos usuarias descritas en el apartado anterior (**Apartado 7.2**) y dejar que interactuaran con ella de forma libre e independiente, se les pidió completar el cuestionario. Los resultados obtenidos se recuperan desde el servicio *Google Forms*, mostrando las estadísticas de la Figura 159.

1 - Indique la dificultad de manejo táctil de la aplicación



Muy fácil.	1	50%
Fácil.	1	50%
Normal.	0	0%
Difícil.	0	0%
Muy difícil.	0	0%

2 - ¿Le ha resultado sencillo decidir qué hacer en cada momento para conseguir lo que deseaba dentro de la aplicación?



No, ha resultado imposible.	0	0%
No, he encontrado gran dificultad.	0	0%
Sí, aunque con cierta dificultad.	0	0%
Sí, ha resultado sencillo.	1	50%
Sí, ha resultado muy sencillo.	1	50%

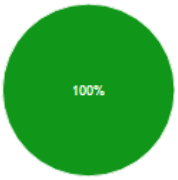
3 - ¿Considera intuitivo el proceso de selección de paciente?

Nada intuitivo.	0	0%
Algo intuitivo.	0	0%
Intuitivo.	0	0%
Bastante intuitivo.	0	0%
Muy intuitivo.	2	100%

4 - ¿Considera intuitivo el proceso de registro en la aplicación?

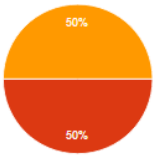
Nada intuitivo.	0	0%
Algo intuitivo.	0	0%
Intuitivo.	0	0%
Bastante intuitivo.	0	0%
Muy intuitivo.	2	100%

5 - ¿Puede realizarse de forma adecuada el seguimiento del paciente?



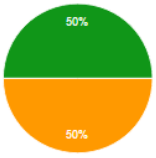
No, nada adecuada.	0	0%
Poco adecuada.	0	0%
Adecuada.	0	0%
Bastante adecuada.	2	100%
Muy adecuada.	0	0%

6 - A la hora de interpretar las gráficas estadísticas, ¿qué grado de dificultad ha encontrado?



Muy fácil.	0	0%
Fácil.	1	50%
Normal.	1	50%
Difícil.	0	0%
Muy difícil.	0	0%

7 - ¿Considera de utilidad la información proporcionada por las gráficas estadísticas?



Nada útil.	0	0%
Poco útil.	0	0%
Útil.	1	50%
Bastante útil.	1	50%
Muy útil.	0	0%

8 - ¿Considera apropiado el envío de correos electrónicos con la información del progreso del paciente en la terapia cognnitiva?

No, nada apropiado.	0	0%
Poco apropiado.	0	0%
Apropiado.	0	0%
Algo apropiado.	0	0%
Muy apropiado.	2	100%

9 - ¿Considera apropiado el envío de correos electrónicos con el cálculo estimado del grado de Alzheimer del paciente?

No, nada apropiado.	0	0%
Poco apropiado.	0	0%
Apropiado.	0	0%
Algo apropiado.	0	0%
Muy apropiado.	2	100%

10 - ¿Considera necesario ampliar la información contenida en los correos electrónicos que reportan el progreso de la terapia cognitiva del paciente?

No, nada necesario.	0	0%
Algo necesario.	2	100%
Necesario.	0	0%
Bastante necesario.	0	0%
Muy necesario.	0	0%

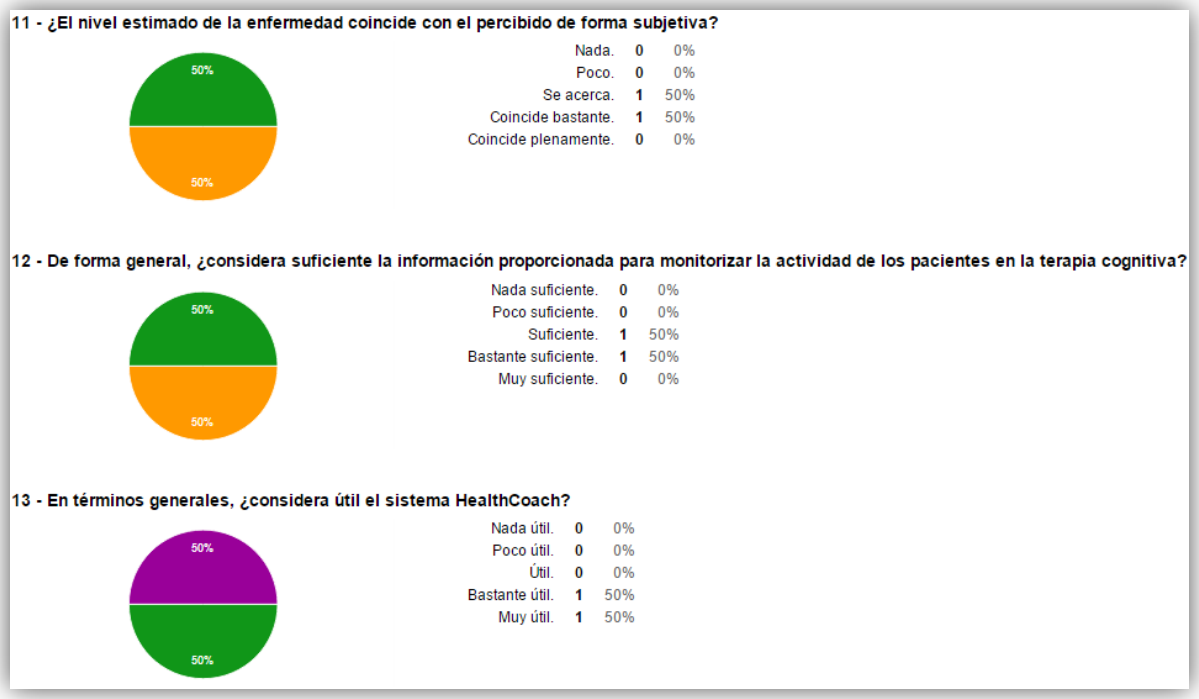


Figura 159. Resultados de encuesta de evaluación de HealthCoach Admin

Analizando los resultados estadísticos obtenidos a partir de las respuestas proporcionadas, se extraen las siguientes reflexiones:

- La interacción táctil con la aplicación no presenta problemas, siendo una interfaz usable y accesible.

- Los procesos de registro de usuario y selección de paciente a monitorizar se realizan sin ningún inconveniente, considerándolos muy intuitivos.

- El seguimiento de la terapia cognitiva del paciente se realiza de forma bastante adecuada a través de la aplicación de responsables (HealthCoach Admin), considerando de utilidad la información proporcionada en las gráficas estadísticas generadas, cuya interpretación se realiza sin complicaciones.

- Existe unanimidad en el envío de correos electrónicos al responsable, cuyo fin es proporcionar información sobre el progreso del paciente en la terapia cognitiva y del cálculo del estadio de la enfermedad, considerándolos muy apropiados. Además, no se considera como crítica una ampliación en la información que proporcionan.

- En cuanto al grado estimado de la enfermedad para los pacientes, los usuarios encuestados consideran que se aproxima al estadio percibido de forma subjetiva o diagnosticado clínicamente.

- A modo de conclusión, los usuarios encuestados consideran suficiente la información proporcionada para monitorizar la evolución del paciente, así como resultar de utilidad el sistema HealthCoach desarrollado en este Proyecto Fin de Carrera.

Capítulo 8

Conclusiones y desarrollos futuros

En este capítulo, último de la memoria, se exponen las conclusiones finales obtenidas con la visión general de los hitos realizados en el Proyecto Fin de Carrera, así como las futuras mejoras y líneas de investigación abiertas a partir del desarrollo del sistema HealthCoach.

8.1 Conclusiones

Una vez llegados al final del proyecto, es momento de valorar los resultados obtenidos en contraste con los objetivos marcados en el capítulo introductorio (**Capítulo 1**). Repasando todos estos objetivos se obtienen las siguientes conclusiones:

- **Tecnologías móviles**

  El interesante estudio de las tecnologías móviles ha permitido conocer de primera mano la rápida evolución de los dispositivos y sistemas operativos móviles, hasta el punto de incorporarse en la sociedad como elementos cotidianos. Esta aceptación ha favorecido expandir su uso a diversos sectores, como son el sanitario, comercio o gubernamental. La realización de la comparativa entre las principales plataformas móviles fue determinante en la elección Android como tecnología principal sobre la que desarrollar el proyecto.

- **Plataforma Android**

  La realización del sistema HealthCoach ha proporcionado amplios conocimientos sobre la arquitectura y características del sistema operativo Android, así como conocer las posibilidades que brinda a la hora de desarrollar aplicaciones nativas gracias al SDK proporcionado por Google. Es precisamente esta herramienta y la extensa documentación proporcionada de manera pública y gratuita, las claves para facilitar la labor del programador. Que el desarrollo se realice mediante código Java y XML influye también positivamente en la cantidad de manuales, documentos y ejemplos de código creados por la comunidad y publicados en Internet de forma altruista.

- **Sistema HealthCoach**

  El escenario propuesto en este proyecto, que particulariza en el Alzheimer la idea de realizar una aplicación estándar dedicada al tratamiento de dolencias médicas, marca el enfoque del análisis y diseño realizados en este proyecto. La necesidad de persistir en base de datos la información consumida y generada por el sistema presenta una de las problemáticas del sistema. Android no dispone de un "conector" nativo con las bases de datos usuales (MySQL, Oracle, PostgreSQL, etc.) por lo que se decidió recurrir a un servicio Web implementado mediante una API REST para

realizar esta tarea. La API se desarrolla en lenguaje PHP por la facilidad para crear servicios de ese tipo mediante el framework SLIM, sin embargo, la falta previa de contacto con este lenguaje ha supuesto un esfuerzo adicional. Otra tarea que ha supuesto un reto, es la implementación del sistema de tarjetas terapéuticas presentadas al usuario en Android, implicando un profundo estudio de las clases *Fragment* para representar cada tarjeta y *ViewPager* para permitir la paginación y transición entre ellas, además de implementar un sistema que genera de forma automática la interfaz que se adapta al tipo de ejercicio cognitivo que representa.

La realización de este proyecto ha permitido descubrir la existencia de otras librerías muy útiles, como son JavaMail para realizar envíos de correos electrónicos desde Android y AandroidPlot para la representación de todo tipo de gráficas. La diferencia percibida en el uso de ambas radica en el nivel de abstracción y dificultad de uso, JavaMail requiere muy pocas líneas de código para llevar a cabo su propósito, sin embargo AndroidPlot necesita un mayor desarrollo y conocimiento de la API.

En cuanto a la implementación del sistema multimodal para la interacción entre usuario y aplicación, Android facilita enormemente la creación de interfaces táctiles definidas mediante plantillas XML. La interacción mediante lenguaje natural supone una labor mucho mayor, aun así, el SDK de la plataforma proporciona acceso al reconocedor de voz de Google para procesar el audio en sus servidores o de forma local gracias al modo offline introducido con Android 4.1. En cuanto a los mensajes del sistema generados mediante la síntesis de texto a voz, Android proporciona las clases necesarias para conectar con motores de síntesis externos, eligiendo para este proyecto la síntesis de voz de Google, debido a la calidad del audio y la naturalidad de las voces.

El desarrollo de las aplicaciones Android y la API REST han resultado con diferencia las tareas más laboriosas del proyecto, sin embargo han resultado enriquecedoras y muy amenas.

- **Evaluación**

A través de las opiniones de los usuarios que han probado las aplicaciones Android, recopiladas en el capítulo de evaluación, se deduce una valoración positiva de HealthCoach. Lo consideran un sistema usable, donde la interacción multimodal es plenamente satisfactoria en su variante táctil, aunque encontrando errores puntuales en la interacción oral que no impiden el correcto progreso en la terapia cognitiva. El reconocimiento automático del habla realiza una de las tareas más complejas del sistema, donde un ruido ambiental o una mala pronunciación por parte del usuario pueden alterar el resultado y obtener respuestas que no se corresponden con las proporcionadas. En cuanto a los mensajes de voz generados por la aplicación son perfectamente entendibles, gracias a la naturalidad del motor de síntesis de texto a voz de Google. Los procesos de registro e identificación se realizan sin problemas, aunque la interpretación de las gráficas estadísticas por parte de los pacientes presenta ciertas complicaciones, por lo que quizás sea un elemento del que se deba prescindir para este colectivo de usuarios y presentarlas exclusivamente en la aplicación de responsables. Para concluir, los pacientes han percibido que ejercitan la mente y les gusta el sistema, en cuanto a los responsables, consideran un sistema útil para monitorizar la progresión del paciente dentro de la terapia cognitiva, considerando además que el cálculo del estadio de la enfermedad se aproxima al diagnóstico clínico o percibido.

Para concluir, Android proporciona todos los elementos y herramientas necesarios para construir aplicaciones de lo más variado, amoldándose a desarrollos de todo tipo de complejidad que además

pueden ser distribuidos a un elevado número de usuarios debido a la predominancia de dispositivos Android entre los poseedores de *smartphones*.

8.2 Desarrollos futuros

La finalización del sistema presentado en este proyecto abre el camino hacia la búsqueda de mejoras en sus módulos y la adición de nuevas funcionalidades. A continuación se proponen algunas:

- Crear una página Web para la administración de los ejercicios presentados. Mediante una interfaz intuitiva podrían darse de alta, editarse o eliminar las tarjetas que son utilizadas en la terapia, cumplimentando los campos necesarios (título, instrucciones, respuesta correcta, imagen asociada, etc.). La existencia de este "portal de administración" evitaría definir la información de las tarjetas directamente en base de datos.

- Incorporar tarjetas cognitivas de tipo praxis, como recoge el BCRS, para poder evaluar también el estado funcional del paciente. Por ejemplo, podría existir una tarjeta que indique realizar la acción de atarse los cordones, ponerse un jersey, etc., siendo el responsable quién marque la acción como satisfactoria o no en la aplicación, por lo que este evolutivo supondría una mayor dependencia en el transcurso de la terapia.

- Extender la interacción multimodal a toda la aplicación, es decir, permitir realizar las mismas acciones que serían ejecutadas mediante pulsación en la pantalla pero utilizando comandos de voz. De esa forma, podría navegarse por los menús sin requerir tanta dependencia táctil, lo que sería especialmente útil en pacientes con problemas de pulso, o cualquier tipo de dolencia en las manos.

- De manera complementaria a la anterior mejora, añadir la posibilidad de proporcionar las respuestas a las tarjetas terapéuticas de manera táctil, por ejemplo, mediante un teclado virtual en pantalla. Gracias a este evolutivo se facilitaría el uso de la aplicación a personas con problemas en el habla.

- Introducir funcionalidades útiles para el colectivo al que se dirige el sistema, como por ejemplo un botón de alerta presentado de manera permanente en la interfaz de la aplicación de pacientes que permita avisar a los responsables en caso de necesitar ayuda en la terapia, o sufrir cualquier tipo de percance. El aviso a los responsables llegaría en forma de SMS o correo electrónico.

- Utilizar técnicas de Big Data [89] para explotar los datos generados con las respuestas proporcionadas por los pacientes a las diversas tarjetas terapéuticas, con el propósito de extraer patrones de uso que lleven a algún tipo de conclusión clínica. El sistema HealthCoach guarda todas las respuestas, correctas e incorrectas, por lo que con un alto número de usuarios se generarían cantidades ingentes de información referente a un colectivo con una dolencia común.

- Estudiar las posibilidades que brindan el Internet de las cosas (*IoT*) en la interconexión de datos entre objetos cotidianos o "*wearables*", como por ejemplo los relojes inteligentes con sistema operativo Android [2, pág. 13]. Estos dispositivos podrían presentar el sistema de

tarjetas diseñado en este proyecto, con muy pocas modificaciones, ya que estos dispositivos comienzan a incorporar micrófono y altavoz.

- Optimización de las aplicaciones desarrolladas para su visualización en tabletas de gran tamaño. Android requiere definir *layouts* específicos en función del tamaño, densidad y orientación de la pantalla, con el propósito de obtener una interfaz con la mejor calidad de imagen y disposición de elementos.

- Añadir un soporte multidioma completo. Actualmente, el paciente puede seleccionar el idioma en el que se le presentan las tarjetas, pero no el mostrado en el resto de módulos.

- Implementar un sistema de "escucha continua" en la interacción oral de la terapia cognitiva, facilitando el proceso de respuesta de los pacientes al evitar tener que pulsar el botón que lanza la actividad de reconocimiento de voz.

Anexo A

Servidor Raspberry Pi. Manual de instalación y configuración.

A.1 Introducción

Este capítulo ofrece el manual de instalación y configuración del servidor del sistema **HealhCoach**. Involucra tanto el **sistema operativo** elegido para el dispositivo hardware que realiza las funciones de servidor, ordenador de bajo coste **Raspberry Pi**, como los **componentes** instalados sobre dicho sistema operativo (ver Figura 160):

- Gestor de bases de datos MySQL con la base de datos *proyectofincarrera*.
- Servidor web Apache, que recibe las peticiones HTTP y contiene el siguiente componente.
- Módulo PHP instalado sobre el servidor web, permite alojar la API REST destinada a realizar las comunicaciones entre aplicaciones cliente Android y la base de datos.

Figura 160. Fotografía real del servidor que aloja el sistema HealthCoach

A.2 Sistema operativo Raspbian

Existe una gran variedad de distribuciones basadas en Linux que han sido compiladas y optimizadas para el dispositivo Raspberry Pi. La elección de **Raspbian** [90] se debe al conjunto de características y ventajas que presenta. De licencia libre (GPL) y basado en Debian Wheezy, cuenta con más de 35.000 paquetes de software pre-compilados para conseguir una ejecución optimizada en Raspberry Pi. Desarrollado por un pequeño equipo de programadores entusiastas, cuenta con un desarrollo y soporte muy activo.

A.2.1 Instalación

Para instalar Raspbian en el servidor Raspberry Pi, que carece de almacenamiento propio, es necesario disponer de una tarjeta SD de al menos 2 GB, aunque se recomienda de 4 GB para poder albergar sin problemas los componentes software necesarios y los datos almacenados en las bases de datos. Se requiere también un dispositivo con sistema operativo Linux, Windows o Mac OS para realizar el proceso de volcado de la imagen a la tarjeta SD. Para este manual se indican las instrucciones para un sistema Windows 7, siendo las siguientes:

1. Descargar la última imagen disponible de Raspbian [91]. El archivo .zip descargado contiene la imagen de Raspbian en su interior, por lo que se debe descomprimir dicho fichero para obtener la imagen en formato .img.

2. Para instalar Raspbian en la tarjeta SD se necesita una herramienta de volcado de imágenes a dispositivos de almacenamiento, si se realiza desde un sistema operativo Windows puede recurrirse a Win32DiskImager [92]. Será necesario extraer el ejecutable contenido en el fichero .zip descargado.

3. Insertar la tarjeta SD en la ranura SD del propio ordenador o portátil, o uno externo conectado mediante USB y observamos que letra de unidad le ha sido asignado en el explorador de Windows.

4. Ejecutar la herramienta ***Win32DiskImager*** con permisos de administrador, haciendo clic con el botón derecho del ratón y seleccionando *Run as administrator*.

5. Seleccionar el archivo de imagen Raspbian descargado anteriormente y la letra de la unidad SD. Es importante asegurarse de seleccionar la unidad correcta ya que se destruirán todos los datos que hubiera anteriormente en ese dispositivo.

6. Pulsar en la opción Write y esperar a que el proceso de escritura de la imagen finalice.

7. Por último, puede cerrarse el programa Win32DiskImager y extraer de forma segura la unidad SD.

Una vez se dispone de la tarjeta SD con el sistema operativo Raspbian basta con introducirla en el dispositivo Raspberry Pi. Pero antes de proceder a arrancar el servidor, es necesario conectarlo al

router existente en la red local mediante un cable RJ-45, y alimentarlo con una fuente de al menos 0,7A y conexión micro-USB. Una vez arrancado, podemos acceder a su interfaz conectando un monitor y teclado, o bien mediante una conexión SSH, que es la forma que se explica a continuación.

A.2.2 Conexión SSH

Para realizar una conexión SSH contra el servidor, es necesario en primer lugar conocer la dirección IP que se le ha asignado. Existen dos formas de averiguarlo, mediante la conexión a la interfaz de administración del router y visualizando la IP establecida al servidor (Figura 161), o bien conectando de forma excepcional un monitor y teclado para lanzar el comando ***ifconfig***, que muestra la información de las interfaces de red presentes en el dispositivo.

Figura 161. Dirección IP del servidor mostrada desde la administración del router

Una vez averiguada la IP podemos realizar la conexión mediante un programa que soporte el protocolo SSH desde Windows. En esta ocasión se ha elegido **Putty** [93]. Desde su interfaz, en el campo *Host Name* se introduce la IP (192.168.1.19) y en *Connection type* se elige SSH, que establecerá el puerto 22 automáticamente, como muestra la Figura 162. El siguiente paso es pulsar en el botón *Open* para establecer la conexión.

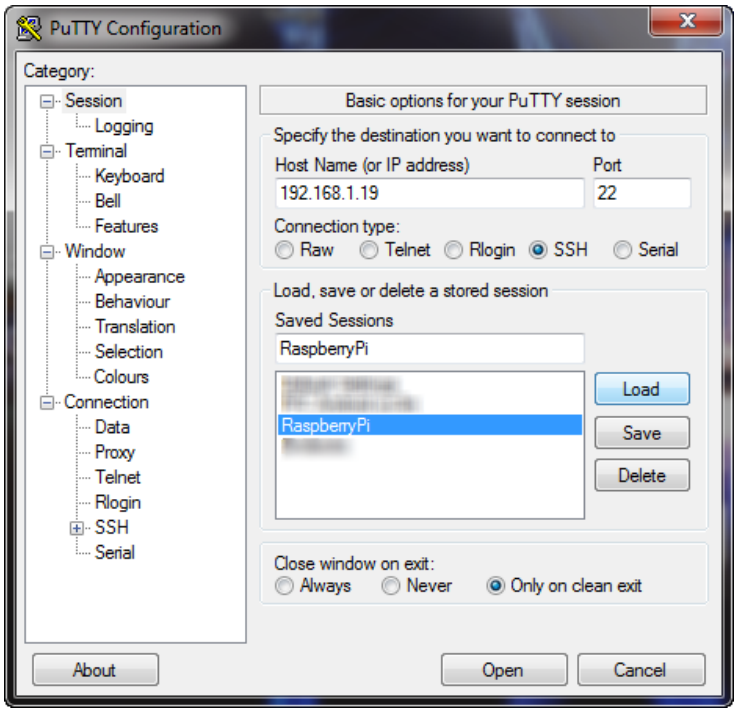


Figura 162. Configuración de acceso al servidor por SSH desde Putty

Se solicitará un usuario y contraseña, que por defecto son "pi" y "raspberry" respectivamente, aunque posteriormente se explica cómo cambiar la contraseña preestablecida. Al introducir los datos de acceso correctamente se obtendrá un terminal en el sistema Raspbian (Figura 163), que permitirá introducir remotamente todo tipo de comandos como si se estuviera físicamente conectado al dispositivo.

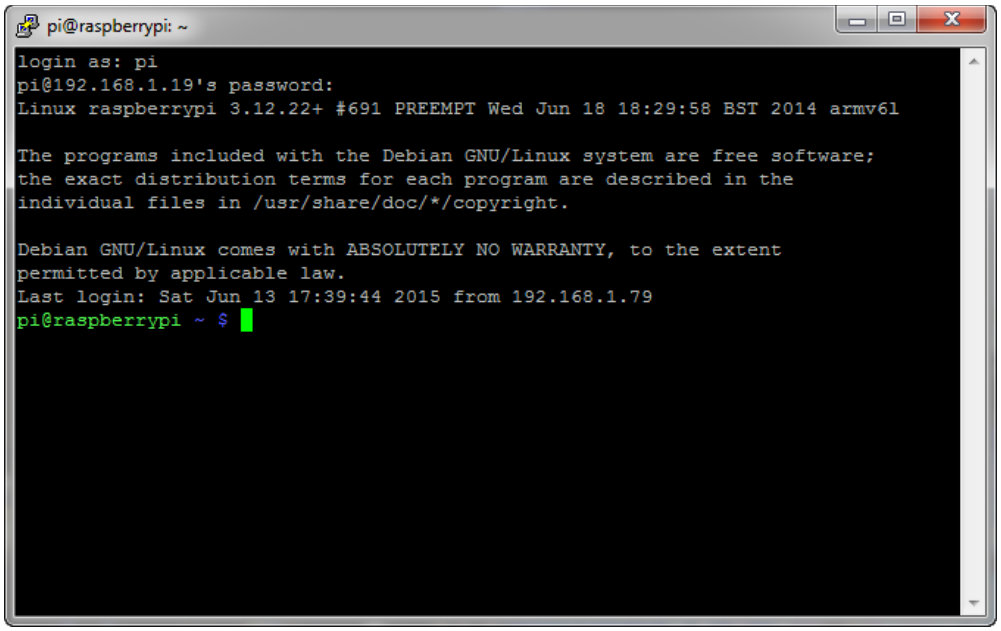


Figura 163. Sesión SSH establecida en Raspbian

A.2.3 Optimización Raspbian

Raspbian incluye una herramienta de configuración para facilitar la modificación de ciertos parámetros relacionados con Raspberry Pi. En la Figura 164 se muestran todas las opciones disponibles, entre ellas la capacidad para expandir el sistema de ficheros al tamaño de la tarjeta SD insertada, cambiar la contraseña por defecto, arrancar directamente en modo escritorio (gráfico) o por línea de comandos, opciones de overclock (modificación de la velocidad y/o voltajes por defecto del procesador y GPU), etc. Para acceder a la herramienta, desde la consola ejecutamos el siguiente comando:

```
sudo raspi-config
```

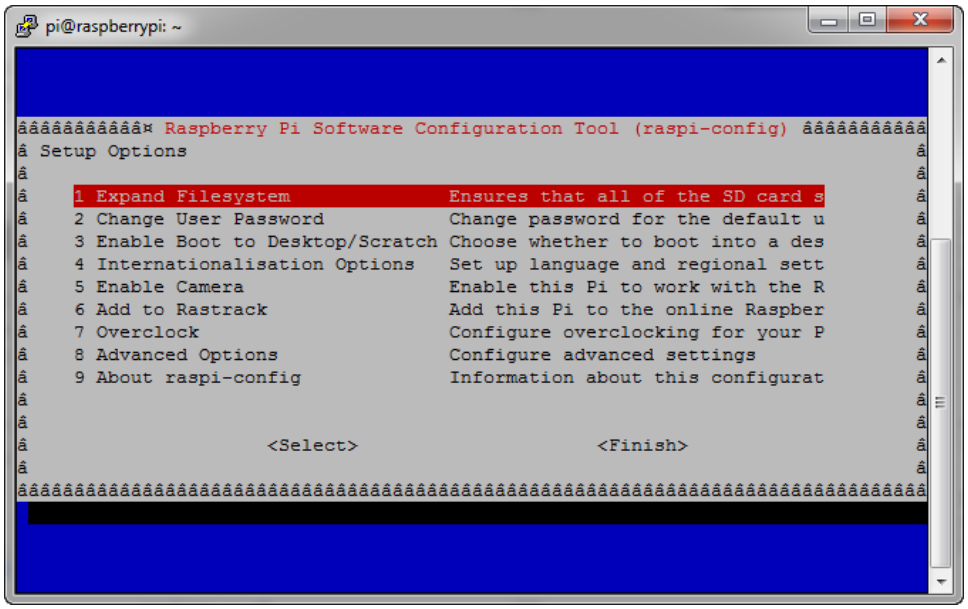


Figura 164. Opciones disponibles en la herramienta de configuración raspi-config

El modelo B de Raspberry Pi posee únicamente 512MB de RAM a compartir entre el sistema y la GPU. Al utilizarse el dispositivo como servidor no es necesario arrancar el entorno gráfico, que por otro lado consumiría más recursos hardware de la máquina. Por lo tanto, entre todas las opciones disponibles resulta interesante minimizar la cantidad asignada al sistema gráfico. Para acceder a ese parámetro, es necesario seleccionar la opción *Advanced Options* (Figura 164) y después *Memory Split*. De entre las opciones que se presentan a modo de ejemplo, se establece un valor de 32 (Figura 165), que representan los MB a disposición de la GPU.

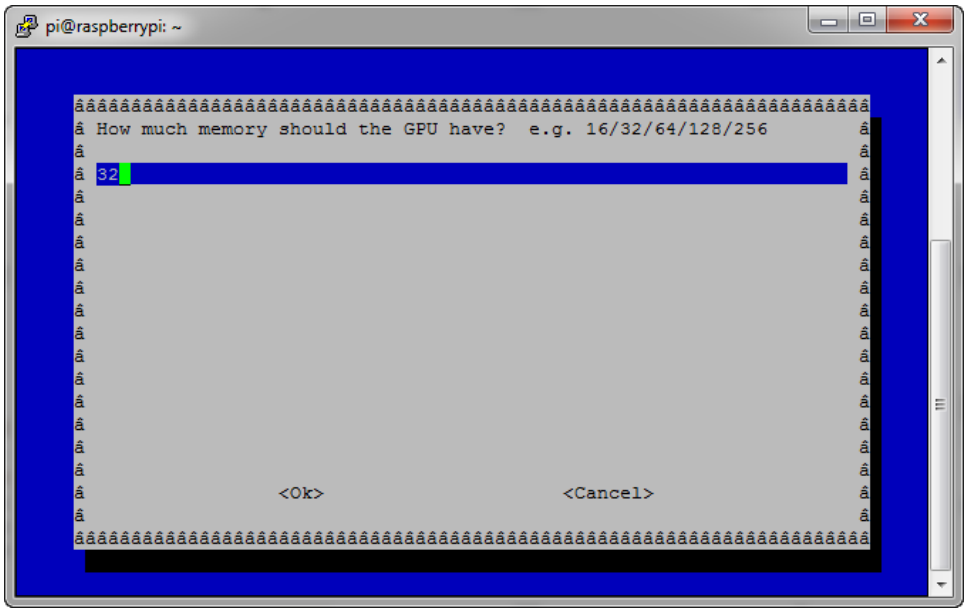


Figura 165. Cantidad de memoria asignada a la GPU del sistema

Una vez realizadas las modificaciones, se selecciona la opción *Finish* y desde el terminal se ejecuta el siguiente comando para reiniciar el sistema:

```
sudo reboot
```

A.3 Servidor de aplicaciones Apache

Para descargar e instalar el servidor de aplicaciones Apache, es necesario conectarse al dispositivo Raspberry Pi mediante SSH, como se muestra en el Apartado A.2.2 de este anexo. Una vez identificados en el sistema Raspbian y desde la línea de comandos, se recurre al sistema de gestión de paquetes (APT) para realizar la descarga local y posterior instalación del servidor Apache mediante el comando ***apt-get***:

```
sudo apt-get install apache2 apache2-doc apache2-utils
```

En la Figura 166 se aprecia la salida obtenida tras la ejecución del comando, donde se conecta a los repositorios de Raspbian para obtener la última versión disponible de Apache en ese momento. Al final de la operación se aprecia un error al recargar la configuración del servidor web: "*Reloading web server config: apache2apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1 for ServerName*".

```
pi@raspberrypi ~ $ sudo apt-get install apache2 apache2-doc apache2-utils
Reading package lists... Done
Building dependency tree
Reading state information... Done
apache2-utils is already the newest version.
apache2-utils set to manually installed.
The following NEW packages will be installed:
  apache2 apache2-doc
0 upgraded, 2 newly installed, 0 to remove and 23 not upgraded.
Need to get 1,775 kB of archives.
After this operation, 12.8 MB of additional disk space will be used.
Do you want to continue [Y/n]? y
Get:1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main apache2 armhf 2.2.22-13+deb7u2 [1,440 B]
Get:2 http://mirrordirector.raspbian.org/raspbian/ wheezy/main apache2-doc all 2.2.22-13+deb7u2 [1,774 kB]
Fetched 1,775 kB in 4s (419 kB/s)
Selecting previously unselected package apache2.
(Reading database ... 78845 files and directories currently installed.)
Unpacking apache2 (from .../apache2_2.2.22-13+deb7u2_armhf.deb) ...
Selecting previously unselected package apache2-doc.
Unpacking apache2-doc (from .../apache2-doc_2.2.22-13+deb7u2_all.deb) ...
Setting up apache2 (2.2.22-13+deb7u2) ...
Setting up apache2-doc (2.2.22-13+deb7u2) ...
[....] Reloading web server config: apache2apache2: Could not reliably determine the server's fully qualified domain name, using 12
. ok
pi@raspberrypi ~ $
```

Figura 166. Ejecución y resultado del comando para la instalación

Para solucionarlo, es necesario editar el fichero *apache2.conf* localizado en la ruta */etc/apache2* y establecer el nombre de servidor a *localhost* mediante la siguiente entrada:

```
ServerName localhost
```

En la Figura 167 se aprecia de forma gráfica este cambio, para realizarlo se ha recurrido al editor de texto GNU nano, disponible dentro de la distribución Raspbian.

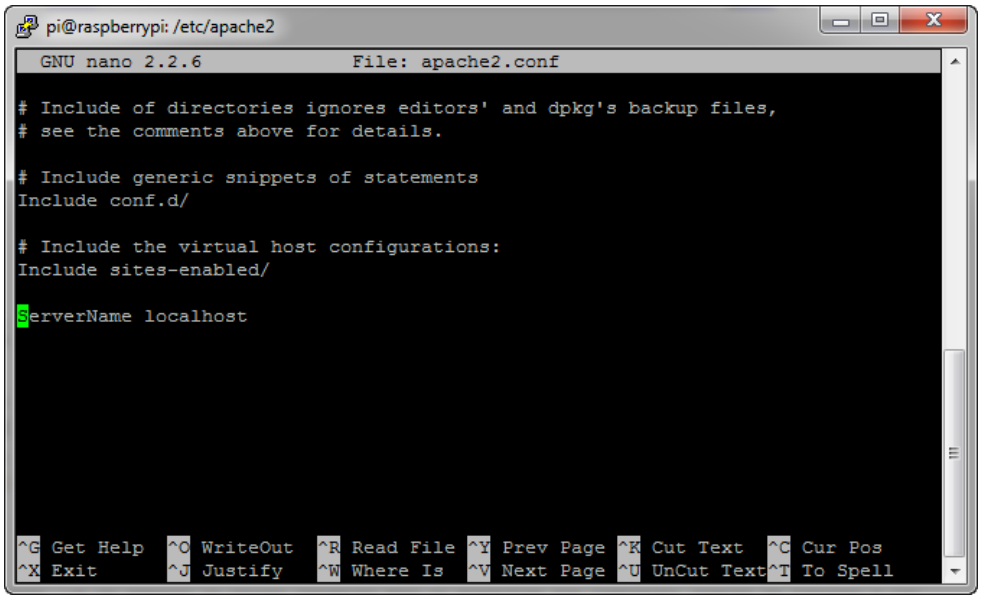


Figura 167. Configuración de fichero apache2.conf

Una vez modificado el fichero, se reinicia el servidor de aplicaciones (Figura 168) con el siguiente comando:

```
sudo /etc/init.d/apache2 restart
```

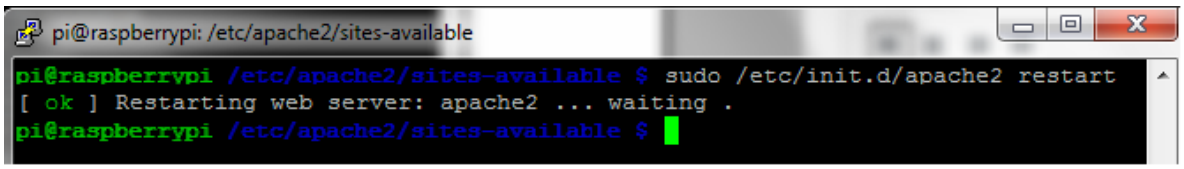


Figura 168. Comando de reinicio del servidor Apache

Por defecto, el servidor Apache escuchará las peticiones a través del puerto 80, pero debido a la existencia de otro servidor ocupando ese puerto, se han modificado las propiedades para que escuche el **puerto 8080**. Se consigue mediante la edición del fichero ***ports.conf***, con el siguiente comando:

```
sudo nano /etc/apache2/ports.conf
```

Modificando los siguientes valores:

```
Listen 80

NameVirtualHost *:80
```

Por estos otros:

```
Listen 8080

NameVirtualHost *:8080
```

La Figura 169 ilustra la modificación en el fichero *ports.conf*.

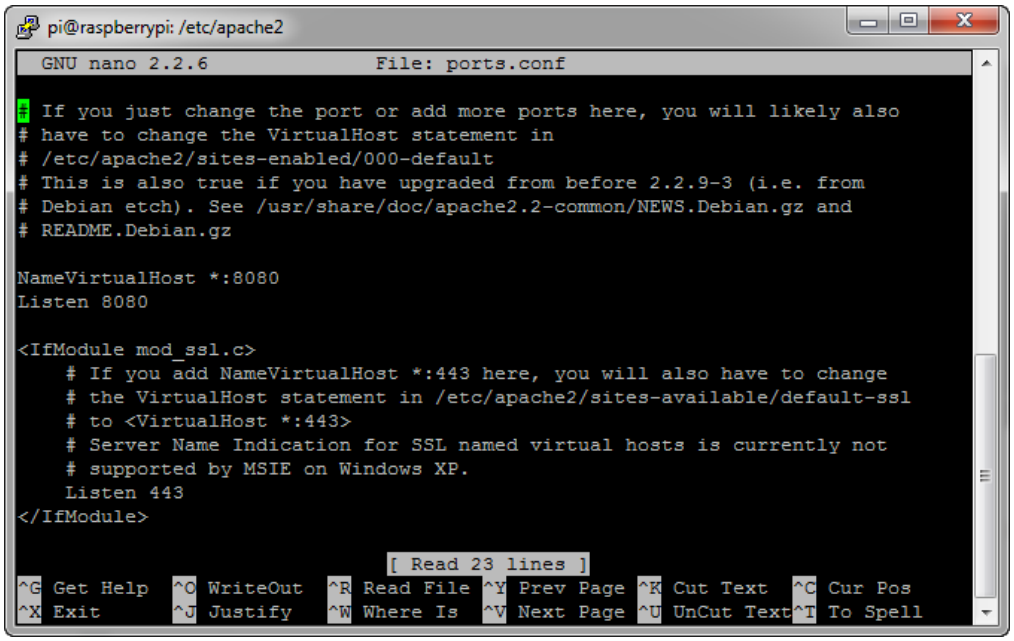


Figura 169. Configuración del fichero ports.conf de Apache

También es necesario modificar el fichero default de Apache para cambiar el puerto del host virtual. Se logra mediante el siguiente comando:

```
sudo nano /etc/apache2/sites-available/default
```

Cambiando esta entrada:

```
<VirtualHost *:80>
```

Por esta otra:

```
<VirtualHost *:8080>
```

Quedando el contenido del fichero como muestra la Figura 170.

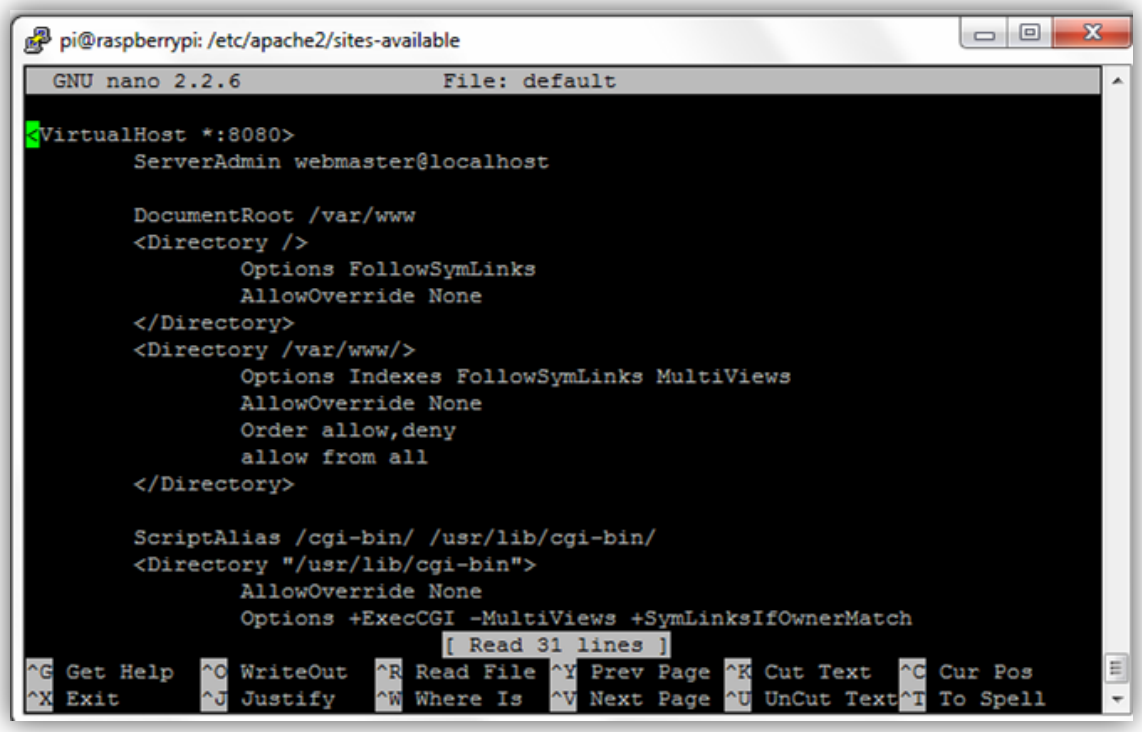


Figura 170. Configuración del puerto de escucha para los VirtualHost

A continuación es necesario reiniciar el servidor de aplicaciones para que surjan efecto los cambios, para ello se ejecuta el comando habitual:

```
sudo /etc/init.d/apache2 restart
```

Para comprobar que el servidor Apache arranca correctamente, se abre un navegador web y se introduce la dirección IP que posee el servidor que aloja la instancia de Apache, así como el puerto en el que realiza la escucha el servidor de aplicaciones, en este caso *192.168.1.19:8080*. La Figura 171 muestra un acceso exitoso al servidor.

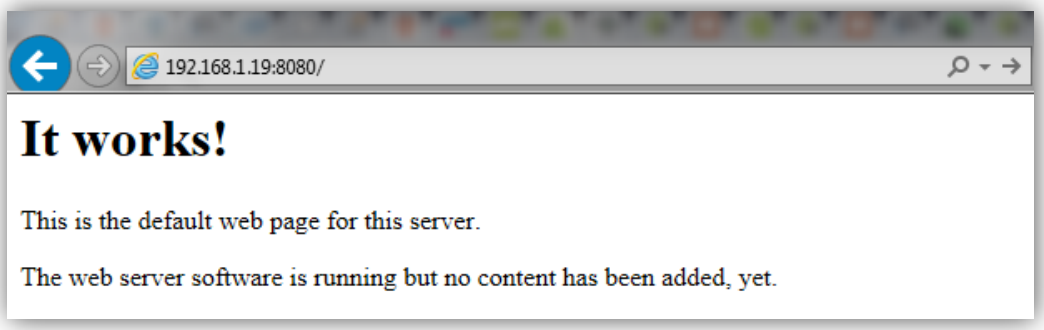


Figura 171. Acceso mediante dirección IP muestra página servida por defecto en Apache

Otra forma de comprobar la conectividad del servidor es a través del comando ***netstat***, que permite visualizar conexiones entrantes y salientes activas en el servidor. A continuación, se especifica el comando y los parámetros necesarios, y la Figura 172 muestra la salida obtenida:

```
sudo netstat -tulpn | grep :8080
```

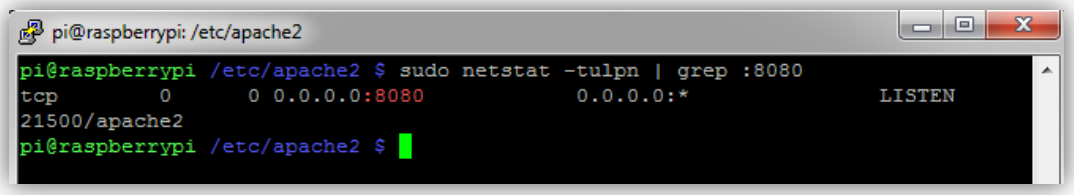


Figura 172. Uso del comando netstat para comprobar la conectividad del servidor Apache

Para utilizar los archivos ***.htaccess*** para que delimitan el acceso a las páginas alojadas en el servidor, será necesario activar el módulo de Apache mod_rewrite:

```
sudo a2enmod rewrite

sudo service apache2 restart
```

En la siguiente página de apache se encuentra toda la información referente a la configuración de los ficheros *.htaccess* [94].

A modo de ejemplo, se muestra una configuración estándar para dicho fichero en la Figura 173.

```
SetEnv APPLICATION_ENV development

RewriteEngine On

RewriteCond %{REQUEST_FILENAME} -s [OR]

RewriteCond %{REQUEST_FILENAME} -l [OR]

RewriteCond %{REQUEST_FILENAME} -d

RewriteRule ^.*$ - [NC,L]

RewriteRule ^.*$ index.php [NC,L]
```

Figura 173. Fichero .htaccess estándar

A.3.1 Acceso al servidor desde red externa

Por defecto, el servidor Apache únicamente permite conexiones procedentes de su misma red. Para permitir que las aplicaciones Android desarrolladas puedan recuperar los datos necesarios para el correcto funcionamiento de la aplicación se realizan dos acciones.

La primera consiste en editar el fichero *default* de Apache, localizado en la ruta */etc/apache2/sites-available/default*. En él se modifican todos los valores **AllowOverride**, que por defecto estarán a valor **None**, por el nuevo valor **All**. Con ello conseguiremos el acceso al servidor de aplicaciones desde el exterior de la red.

La segunda medida consiste en utilizar el servicio gratuito de resolución de nombres de dominio a dirección IP dinámica, llamado **No-IP**. Este servicio proporciona un nombre de dominio del estilo *xxxx.noip.me* asociado a la IP dinámica existente en el router. Una vez completado el registro en su página web [95], es necesario acceder a la sección llamada **Hosts/Redirects** y elegir la opción **Add Host**, donde se elige el *Hostname (nombre de equipo)* y el dominio de entre una lista de disponibles. El segundo campo a rellenar es el *Host Type*, que deberá establecerse como *DNS Host(A)*. La dirección IP se rellenará automáticamente con la dirección pública del router. La opción *Assign to Group* no es necesario rellenarla. Si el *hostname* se encuentra disponible, se creará satisfactoriamente mostrando la siguiente pantalla de la Figura 174.

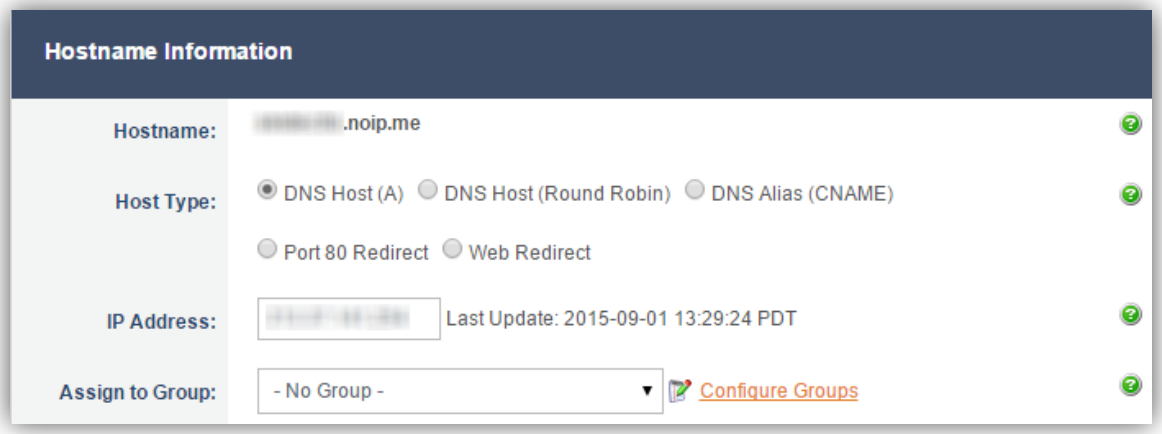


Figura 174. Configuración de nuevo equipo No-IP

Como puede observarse en la figura anterior, en el campo *IP Address* aparece la fecha de última actualización, se debe a que es necesario algún mecanismo que informe al servicio No-IP cuando se realiza un cambio en la IP pública del router. Una de las maneras más sencillas de actualizar esta información es utilizando la propia interfaz de configuración del router para configurar los servicios de DNS dinámicos. La Figura 175 muestra los parámetros necesarios para realizar la asociación, como son el proveedor de DNS dinámico, el correo electrónico con el que se realizó el registro en dicho servicio y la contraseña asociada, para finalizar con el nombre de equipo.

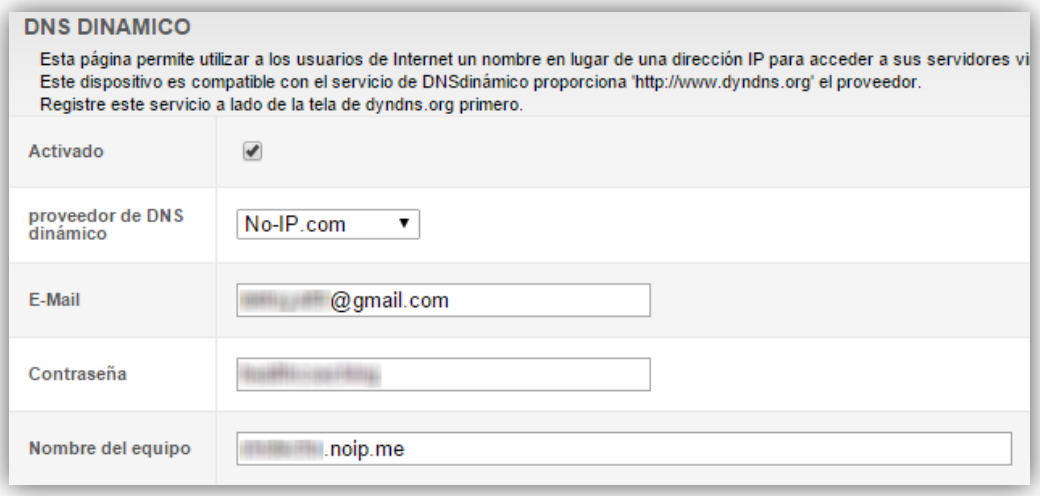


Figura 175. Asociación de cuenta No-IP en el router

El último paso implica crear una regla de redirección desde el router, que normalmente puede crearse desde la administración avanzada del mismo. En la ruta se establece la IP estática del servidor para indicar que cualquier llamada externa al puerto 80 de la dirección pública del router sea redirigida al puerto 8080 del servidor, como se muestra en la Figura176.

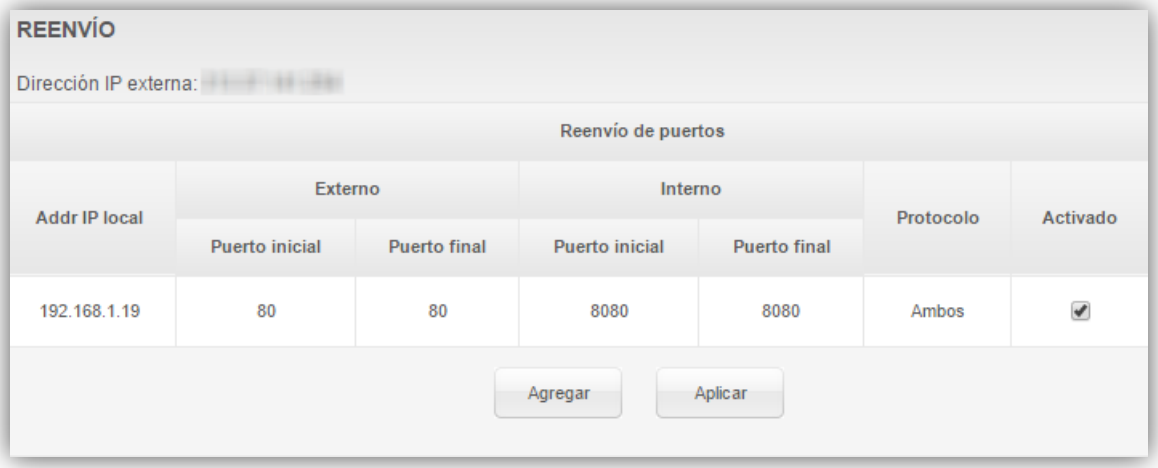


Figura 176. Regla de redirección al servidor del sistema

Una vez se configure la cuenta ***No-IP*** para realizar la resolución de nombres podrá invocarse directamente utilizando la url *xxxxx.noip.me,* obteniendo el resultado mostrado en la Figura 177.

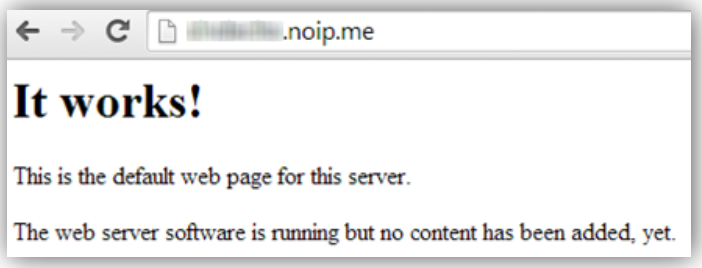


Figura 177. Acceso a Apache mediante dominio No-IP

A.4 Módulo PHP

Para la instalación del módulo PHP, integrado en la instancia del servidor de aplicaciones Apache que permite servir páginas de ese lenguaje, se utiliza el comando *apt-get* indicando los siguientes parámetros:

```
sudo apt-get install libapache2-mod-php5 php5 php-pear php5-xcache
```

También es necesario el conector con el gestor de bases de datos MySQL, por lo que se realiza su instalación con el comando:

```
sudo apt-get install php5-mysql
```

El resultado de la instalación se muestra en la Figura 178.

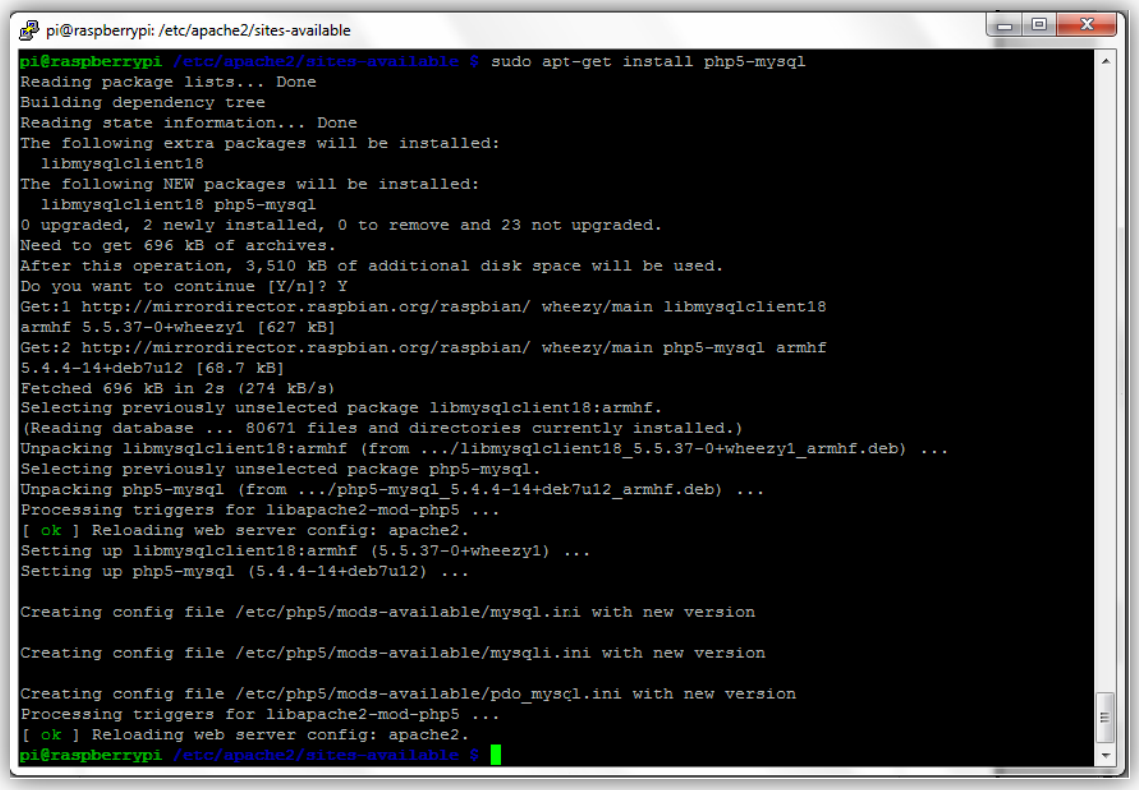


Figura 178. Instalación de PHP5 por línea de comandos

El siguiente paso es realizar una prueba que confirme si se pueden servir páginas PHP con la instalación de este nuevo módulo. Para ello, es necesario posicionarse en el directorio ***/var/www*** donde se alojan por defecto las páginas a servir por Apache:

```
cd /var/www
```

A continuación se crea la carpeta *prueba* y se accede a ella:

```
sudo mkdir prueba

cd prueba
```

Una vez dentro se crea el fichero **index.php**:

```
sudo touch index.php
```

Y se edita mediante un editor de texto, en este caso se utilizará Nano:

```
sudo nano index.php
```

Añadiendo el contenido mostrado en la Figura 179.

```
<?php
class RedeemAPI {
    // Main method to redeem a code
    function redeem() {
        echo "Hello, PHP!";
    }
}
// This is the first thing that gets called when this page is loaded
// Creates a new instance of the RedeemAPI class and calls
the redeem method
$api = new RedeemAPI;
$api->redeem();
?>
```

Figura 179. Código PHP de ejemplo para imprimir un texto

Este sencillo fichero PHP crea una instancia de la clase RedeemAPI para invocar un método encargado de mostrar por pantalla el texto "Hello, PHP!". El resultado de su invocación puede comprobarse accediendo mediante un navegador web a la url *192.168.1.19:8080/prueba* o *192.168.1.19:8008/prueba/index.php*.

Otra manera de comprobar si el servidor Apache gestiona correctamente las páginas PHP es por medio de la utilidad curl. Para instalarla basta con ejecutar el comando mostrado en la Figura 180.

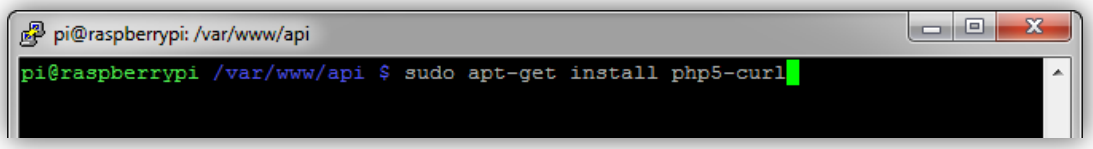


Figura 180. Instalación de la herramienta curl

Una vez instalado se ejecuta el comando:

```
sudo curl http://192.168.1.19:8080/prueba/
```

El acceso al servidor se muestra en la Figura 181.

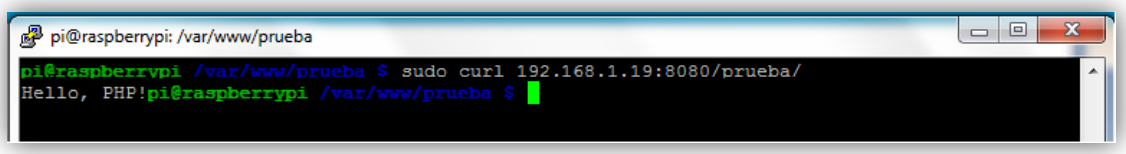


Figura 181. Prueba de acceso a ficheros PHP a través de comando curl

El siguiente paso es probar la conectividad del servicio con la base de datos MySQL mediante el código PHP contenido en el fichero *ejemplo.php* (Figura 182).

```
<?php

$username = "admin";

$password = "admin";

$hostname = "192.168.1.19";


//connection to the database

$dbhandle = mysql_connect($hostname, $username, $password)

 or die("Unable to connect to MySQL");

echo "Connected to MySQL<br>";


//select a database to work with

$selected = mysql_select_db("proyectofincarrera",$dbhandle)

  or die("Could not select examples");


//execute the SQL query and return records

$result = mysql_query("SELECT IDU, NAME, AGE FROM USERS");


//fetch the data from the database

while ($row = mysql_fetch_array($result)) {

   echo "IDU:".$row{'IDU'}." Nombre:".$row{'NAME'}." Edad: ".$row{'AGE'}."<br>";

}
```

```
//close the connection

mysql_close($dbhandle);

?>
```

Figura 182. Acceso a base de datos MySQL desde código PHP

Este fichero está disponible a través de la ruta *http://192.168.1.19:8080/prueba/ejemplo.php*. Como puede apreciarse en el código, realiza la conexión con la base de datos MySQL indicando el usuario, contraseña, dirección del servidor y nombre de la base de datos seleccionada, para posteriormente recuperar un listado con los identificadores y nombres de usuarios disponibles en la tabla *USERS*.

La Figura 183 ilustra el resultado de la invocación de esa URL desde un navegador web.

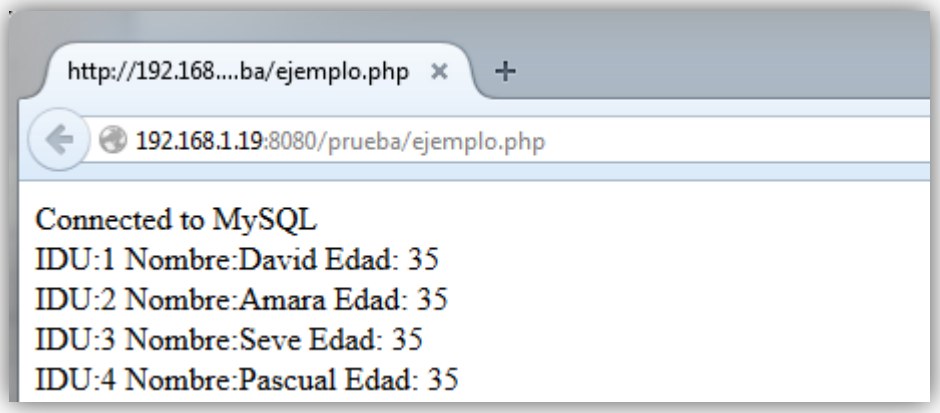


Figura 183. Resultado de acceso a base de datos desde PHP

A.5 Servidor de bases de datos MySQL

Para la instalación del servidor de base de datos MySQL se recurre, como es habitual, al comando apt-get de la siguiente forma:

```
sudo apt-get install mysql-server
```

El asistente de instalación preguntará por un nombre de usuario y contraseña. En esta ocasión se ha establecido *admin* como nombre de acceso. Para acceder a MySQL será necesario arrancar el servicio y realizar la conexión indicando como parámetros el usuario y la dirección donde se encuentra:

```
sudo service mysql start

mysql –u admin –h localhost –p
```

Acompañando la opción –u se indica el usuario, junto a –h el nombre de máquina o dirección IP donde se encuentra el servidor MySQL y –p indica que se requiera la contraseña por terminal para que no sea visible en el momento de introducirla (Figura 184).

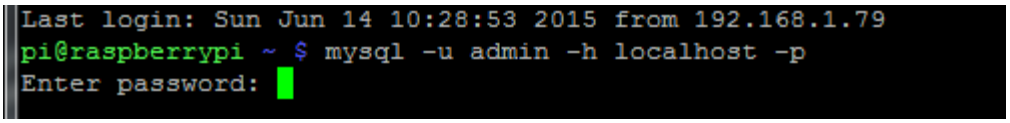


Figura 184. Acceso a instancia mysql desde línea de comandos

Una vez dentro pueden consultarse todas las bases de datos presentes en el servidor:

```
show databases;
```

También pueden listarse las tablas existentes en la base de datos deseada, por ejemplo para *proyectofincarrera*:

```
use proyectofincarrera;

show tables;
```

Obteniendo el resultado mostrado en la Figura 185.

```
pi@raspberrypi ~ $ mysql -u admin -h localhost -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 40
Server version: 5.5.37-0+wheezy1 (Debian)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use proyectofincarrera;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+------------------------------+
| Tables_in_proyectofincarrera |
+------------------------------+
| CARDS                        |
| RECORDS                      |
| USERS                        |
+------------------------------+
3 rows in set (0.00 sec)
```

Figura 185. Tablas presentes en la base de datos proyectofincarrera

Una vez dentro del servidor MySQL, es importante asignar privilegios al usuario de base de datos para que pueda ejecutar cualquier tipo de sentencia SQL (insert, udpate, delete, etc.). A modo de ejemplo se muestran los comandos SQL utilizados para crear el usuario admin y asignarle privilegios plenos (Figura 186).

```
CREATE USER admin@'localhost' IDENTIFIED BY admin;

GRANT ALL PRIVILEGES ON *.* TO admin@'localhost' WITH GRANT OPTION;

CREATE USER admin@'%' IDENTIFIED BY admin;

GRANT ALL PRIVILEGES ON *.* TO admin@'%' WITH GRANT OPTION;
```

Figura 186. Sentencias SQL para creación de usuario y asignación de privilegios

En el fichero de configuración localizado en la ruta *etc/mysql/my.cnf* existen dos importantes propiedades que deberán ser comprobadas:

- **port**. Su valor por defecto es 3306, puede especificarse otro pero es esencial conocer el puerto para realizar la conexión con la base de datos.

- **bind-adress**. Por defecto 127.0.0.1, indica que únicamente se aceptarán conexiones a la base de datos desde la propia máquina, para aceptar desde el exterior es necesario comentar esta línea.

Una vez realizadas las configuraciones anteriores es necesario reiniciar el servicio mysql:

```
sudo service mysql restart
```

A partir de este punto debería ser posible la conexión remota a la base de datos. A modo de ejemplo, en la realización de este proyecto se ha utilizado ***Toad for MySQL*** (Figura 187) indicando los datos:

- Host: 192.169.1.19
- User: admin
- Database: proyectofincarrera
- Port: 3306

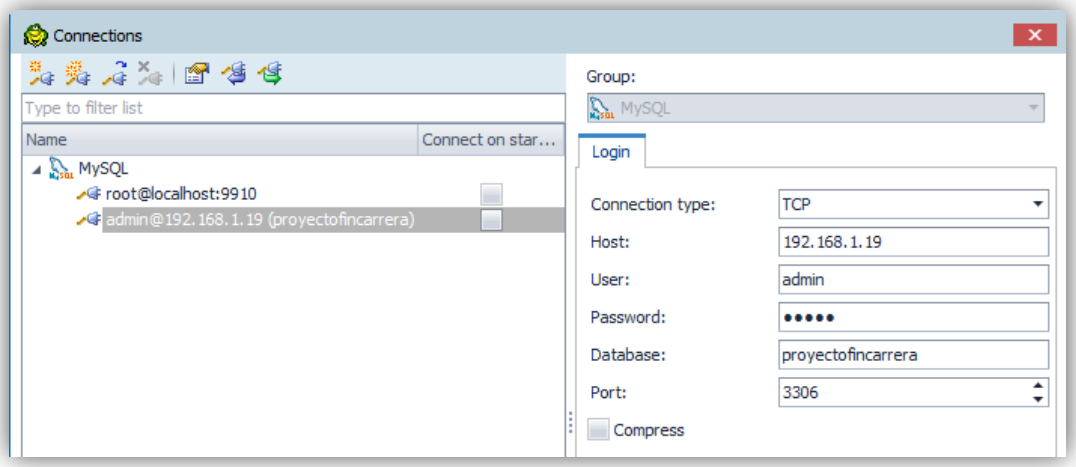


Figura 187. Parámetros de acceso para conexión remota a la base de datos

Anexo B

Manual de usuario

El objetivo de este anexo es presentar de forma gráfica la funcionalidad de las dos aplicaciones desarrolladas, utilizando para ello imágenes reales de la interfaz de usuario, caracterizada por un diseño sencillo e intuitivo. Cada apartado presentado a continuación se corresponde con una función concreta a disposición de los usuarios.

B.1 App pacientes

En primer lugar se muestran las pantallas de la aplicación destinada a los pacientes que realizan la terapia cognitiva. Como puede observarse en el **Apartado B.2**, esta aplicación y la de responsables comparten buena parte de la funcionalidad y por tanto de la interfaz de usuario, aunque se aprecian matices diferenciadores en cada una de ellas.

B.1.1 Inicio

Al pulsar el icono de la aplicación, el usuario observará una pantalla de inicio que muestra el logotipo del sistema **HealthCoach** (Figura 188). Pasado un segundo, esta pantalla se oculta para dar paso a la pantalla de identificación mostrada en el siguiente apartado.

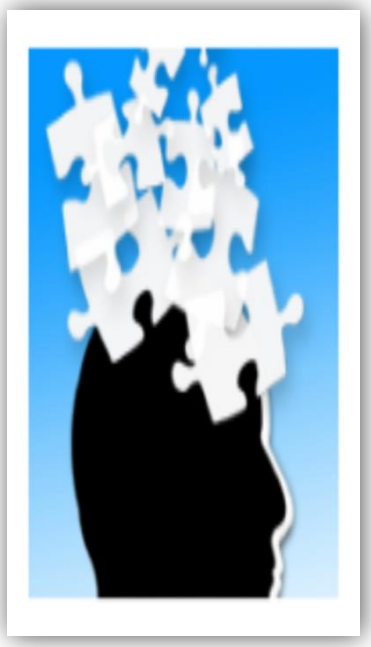


Figura 188. Pantalla inicial con el logotipo de HealhCoach

B.1.1.1 Conexión de red

Una vez mostrado el logo de la aplicación (**Apartado B.1.1**), la aplicación comprueba la existencia de una conexión a Internet en el dispositivo móvil, ya sea a través de Wi-Fi o mediante datos móviles, obligatoria para el correcto funcionamiento de la aplicación. Si no se detecta conectividad, el usuario es informado de esta situación mediante la pantalla mostrada en la Figura 189. Como puede observarse, el usuario es instado a abrir la configuración Wi-Fi de su terminal móvil para habilitar dicha conexión, en caso de rechazar la propuesta la aplicación se cerrará.

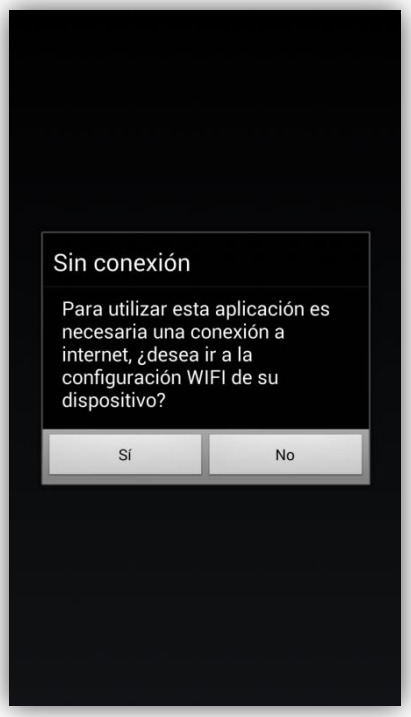


Figura 189. Mensaje informativo de ausencia de conexión de red

B.1.2 Identificación

Esta pantalla cumple el propósito de identificar al usuario dentro del sistema, recuperando su progreso y permitiendo proseguir con la terapia cognitiva en el punto que la dejó además de toda la información relativa a su progreso.

Para realizar el proceso de identificación se presenta una lista con todos los usuarios que han sido registrados anteriormente desde ese mismo dispositivo móvil (Figura190), siendo necesario pulsar en el correspondiente al paciente que va a realizar los ejercicios. Cada componente de la lista muestra una silueta de hombre o mujer en función del sexo, el nombre y primer apellido, nivel actual de las tarjetas de terapia que está realizando y un icono del lenguaje en el que se encuentran dichas tarjetas.

Debajo del listado de usuarios recuperados, existe un enlace a la pantalla de registro identificado por el siguiente texto: "¿Quieres crear otro usuario? ¡Hazlo ahora!". Si es la primera vez que se accede a la aplicación desde el dispositivo móvil, o bien no se recupera ningún usuario asociado al terminal, se navegará automáticamente a la pantalla de registro mostrada en el **Apartado B.1.3**.

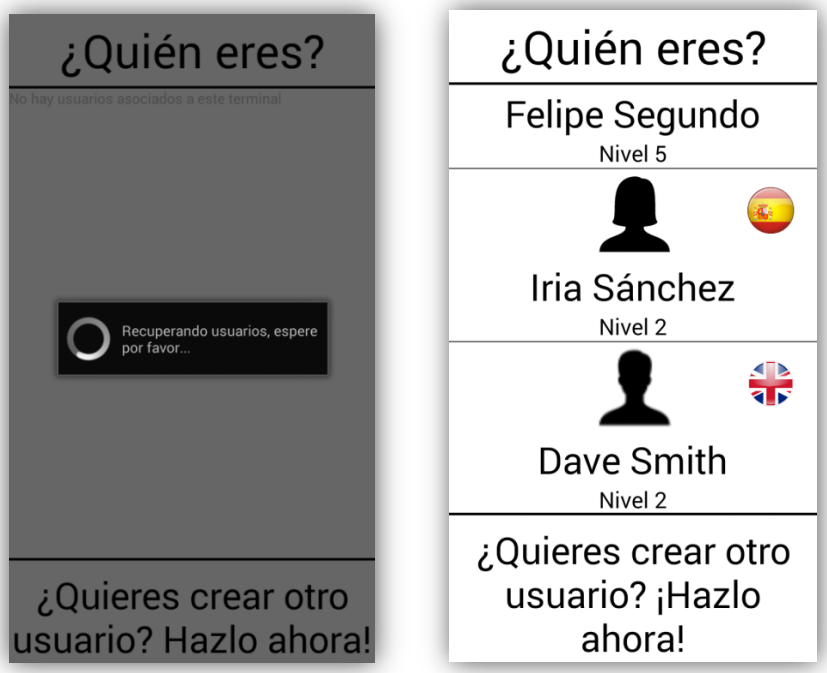


Figura 190. Identificación de paciente de HealthCoach

B.1.3 Registro

El proceso de registro se realiza informando los campos nombre, primer y segundo apellidos, email del responsable, edad, sexo e idioma de las tarjetas de terapia (castellano o inglés). Una vez completados se pulsa en el botón *Enviar* para añadir los datos del usuario en el sistema (Figura 191).

Figura 191. Ejemplo de proceso de registro de paciente

Si se han introducido todos los campos correctamente se navegará a la pantalla de identificación descrita en el **Apartado B.1.2**, donde se observará ya al usuario recién inscrito. Si por el contrario falta algún dato por rellenar, se mostrará un mensaje de error instando al usuario a subsanarlo, como el mostrado en la Figura 192.

Figura 192. Validación del formulario de registro de usuario

B.1.4 Menú principal (Bienvenida)

Una vez identificado al usuario en el sistema, la pantalla de bienvenida muestra el nombre del usuario dentro de un saludo e indica el nivel de dificultad actual de las tarjetas que se le presentan dentro de la terapia cognitiva. La parte central de la pantalla contiene tres enlaces que realizan por orden descendiente las acciones de iniciar o continuar la terapia cognitiva (*Comenzar ;-)*), mostrar el menú de estadísticas de usuario (*Estadísticas*) y cambiar el usuario actual (*Otro usuario*). En la parte inferior se presenta de forma aislada la opción de cerrar la aplicación (*Salir*). Esta pantalla se ilustra en la Figura 193.

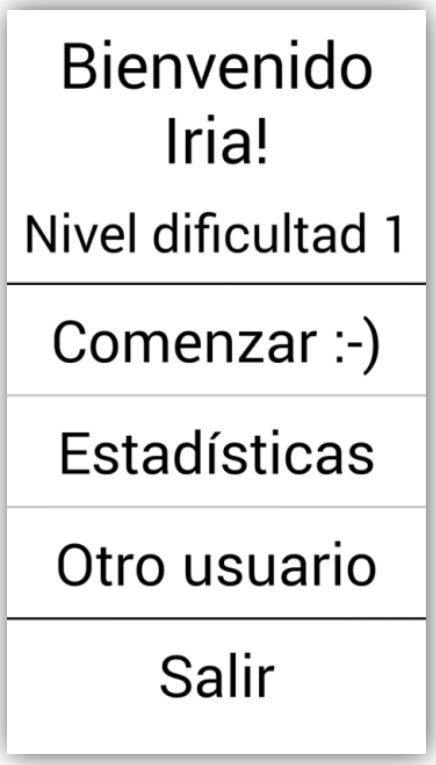


Figura 193. Pantalla de bienvenida con menú principal

B.1.5 Terapia cognitiva

Una vez que el usuario pulsa en *Comenzar :-)* se le presenta el sistema de tarjetas de ejercicios cognitivos que deberá realizar con fines terapéuticos. Si es la primera vez que accede se mostrarán todas las tarjetas del nivel 1 hasta completarlo, mientras que si ya había comenzado la terapia previamente se retomará el punto en que lo dejó, presentando la siguiente tarjeta que le tocaba realizar. En total son 6 los niveles de dificultad existentes.

Las tarjetas presentadas al usuario (Figura 194) disponen de los siguientes elementos:

- En la parte superior se muestra la pregunta o acción que debe realizar el usuario.
- Debajo y de forma opcional, una imagen o texto sobre los que realizar la acción o preguntas anteriores.
- Un botón representado por un icono con la imagen de un micrófono que el usuario debe pulsar para proporcionar la respuesta oral.
- En la parte inferior se muestra la respuesta que proporciona el usuario una vez captada por el reconocedor de voz.

Figura 194. Interacción con tarjetas de terapia cognitiva

El usuario dispone de tres intentos para contestar de forma correcta cada ejercicio. Si proporciona una respuesta errónea o el sistema es incapaz de detectar correctamente la voz, la aplicación comunica al usuario esta situación mediante un mensaje de voz y otro textual, que describen de forma ampliada cuál es la acción que debe llevar a cabo en esa tarjeta.

Si se consumen las tres oportunidades, la aplicación pasa automáticamente a la siguiente tarjeta, siendo informados mediante un mensaje textual y otro sonoro. De forma análoga, en caso de éxito se comunica al usuario el acierto de forma dual, y se navega automáticamente a la próxima tarjeta.

B.1.5.1 Pantalla entre niveles

Cada vez que el usuario completa todas las tarjetas de un nivel, que el algoritmo de adaptabilidad ha considerado previamente que debe realizar, la aplicación muestra al paciente una pantalla que comunica la superación del nivel que se encontraba realizando, como muestra la Figura 195. Desde esta pantalla, el paciente elige a través del menú presentado las opciones de continuar con la terapia en el siguiente nivel, guardar el progreso y salir de la aplicación, o ir al menú de gráficas estadísticas.

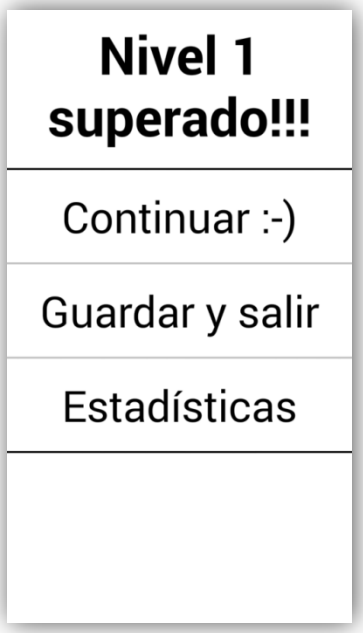


Figura 195. Pantalla informativa de superación de nivel

B.1.5.2 Pantalla finalización terapia

Una vez se han completado todas las tarjetas que el sistema ha establecido para el usuario, se muestra una pantalla que felicita e informa de la finalización de la terapia (Figura 196). Puede llegarse a esta situación por dos motivos:

- Finalización de todos los seis niveles de la aplicación.
- El algoritmo de adaptabilidad, en base a los resultados de los niveles anteriores, descarta ámbitos cognitivos hasta el punto que ya no existe ninguno a realizar.

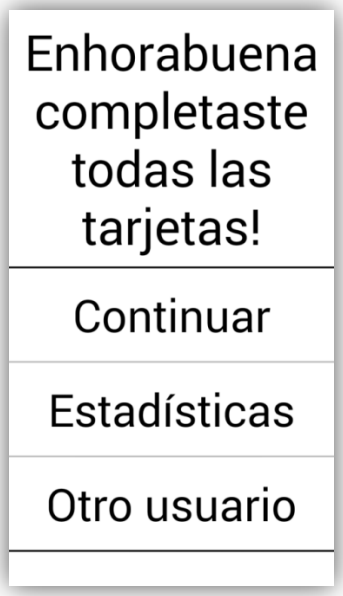


Figura 196. Pantalla de finalización de terapia cognitiva

Llegados a este punto al usuario se le presentan tres opciones, *Continuar* que le lleva a la pantalla de Bienvenida (**Apartado B.1.4**), *Estadísticas* (descrita en el **Apartado B.1.6**) que navega al menú de estadísticas de usuario disponibles, y *Otro usuario* que permite acceder a la pantalla de identificación mostrada en el **Apartado B.1.2**.

B.1.6 Gráficas estadísticas

Existe una pantalla que lista las distintas estadísticas generadas con la evolución de la terapia, es decir, con el progreso del paciente en la realización de las tarjetas cognitivas. El usuario puede seleccionar cualquiera de las tres estadísticas disponibles o pulsar en *Volver*, localizado en la parte inferior de la pantalla, para regresar a la pantalla de bienvenida (**B.1.4 Bienvenida**). La Figura 197 muestra la pantalla que contiene estos elementos.

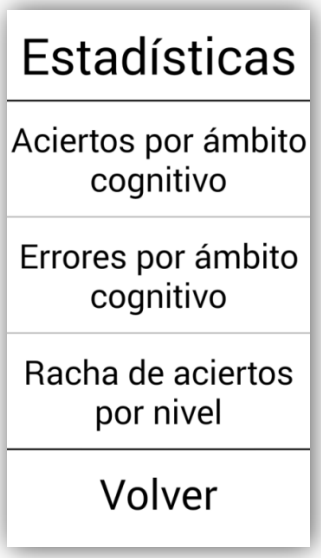


Figura 197. Pantalla de estadísticas

B.1.6.1 Aciertos por ámbito cognitivo

La primera de las estadísticas disponibles muestra los aciertos totales en los ejercicios de la terapia por ámbito cognitivo. Se basa en una gráfica en forma de tarta en la que cada porción recibe un color para facilitar la apreciación de las proporciones. Al pulsar cualquiera de las porciones se muestra un mensaje emergente que indica el número concreto de aciertos dentro de ese ámbito, como puede observarse en la Figura 198.

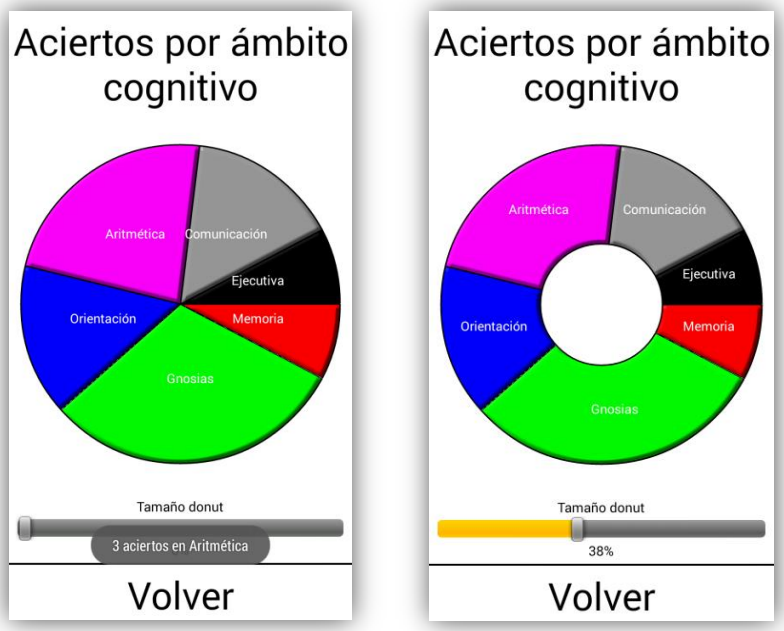


Figura 198. Gráfica estadística en forma de tarta

Adicionalmente, existe una barra con un control deslizable para establecer en tiempo real el tamaño de "donut" asociado a la tarta. En la parte inferior existe un botón "*Volver*" para regresar al menú de gráficas estadísticas mostrado en el **Apartado B.1.6**.

B.1.6.2 Errores por ámbito cognitivo

Esta gráfica muestra las estadísticas asociadas a los errores cometidos en la resolución de las tarjetas en función de su ámbito cognitivo. Como ilustra la Figura 199, la representación se realiza a través de barras situadas en el eje de ordenadas con el número de tarjetas y el de abscisas con los distintos ámbitos cognitivos. Por cada ámbito cognitivo se representa una doble barra (dos series) para indicar tanto el número total de tarjetas como el número de errores sobre ese total. Además, permite pulsar sobre cualquiera de las barras para obtener un mensaje con información aclaratoria.

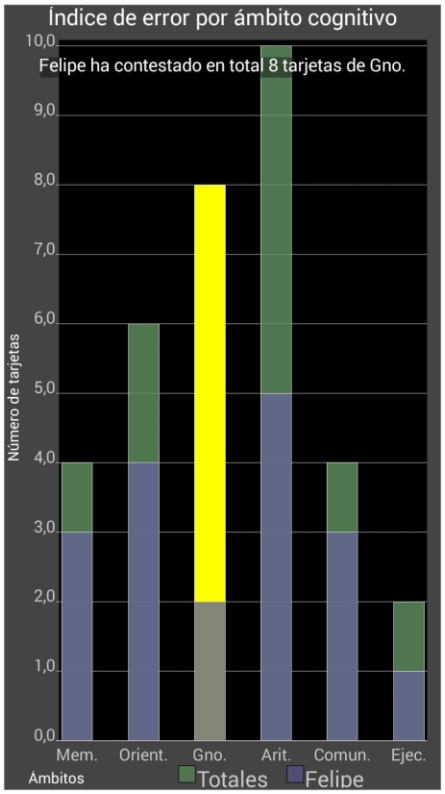


Figura 199. Gráfica estadística de errores por ámbito cognitivo cometidos por un usuario

B.1.6.3 Racha máxima de aciertos

La tercera y última gráfica, proporciona información referente a la racha máxima de aciertos en cada uno de los niveles de la terapia que ha realizado el usuario. Como muestra la Figura 200, el eje de ordenadas indica el número de aciertos alcanzado mientras que el de abscisas representa el nivel de terapia cognitiva.

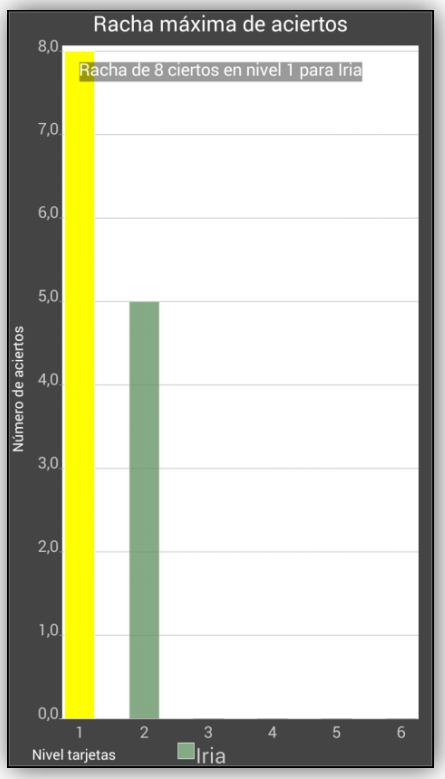


Figura 200. Gráfica estadística de racha máxima de aciertos de usuario

El paciente puede pulsar las barras expuestas para obtener su significado dentro de la gráfica mediante un mensaje emergente.

B.2 App responsables

En este apartado se muestran las pantallas de HealthCoach Admin, la aplicación destinada a los responsables que monitorizan la terapia cognitiva de los pacientes. A pesar de compartir parte de la funcionalidad de la aplicación de pacientes (HealthCoach) cuyas pantallas pueden observarse en el **Apartado B.1**, esta aplicación dispone también de características propias que serán descritas.

B.2.1 Inicio

La pantalla de inicio de la aplicación de responsables es igual a la mostrada anteriormente en el **Apartado B.1.1**.

B.2.2 Identificación

El proceso de identificación para responsables es similar al expuesto para los pacientes en el **Apartado B.1.2**. En un mismo dispositivo pueden registrarse varios responsables, por lo que un listado de usuarios registrados con ese terminal es presentado para que el individuo que accede pueda

seleccionarse. Cada elemento de la lista mostrará una silueta indicando el sexo de la persona, nombre, primer apellido y el número de pacientes a su cargo.

En la parte inferior de la pantalla de identificación existe un botón para realizar la navegación al proceso de registro para nuevos responsables.

Todas estas características pueden observarse en la Figura 201.

Figura 201. Selección de responsable en el terminal móvil

B.2.3 Registro

El proceso de registro para responsables es similar al descrito para los pacientes en el **Apartado B.1.3** con la peculiaridad de que en esta ocasión el email de registro si es el del propio responsable que desea darse de alta en la plataforma. En este registro no es necesario indicar el idioma ya que no realizará ninguna interacción con las tarjetas terapéuticas. Se puede apreciar esta pantalla de registro en la Figura 202.

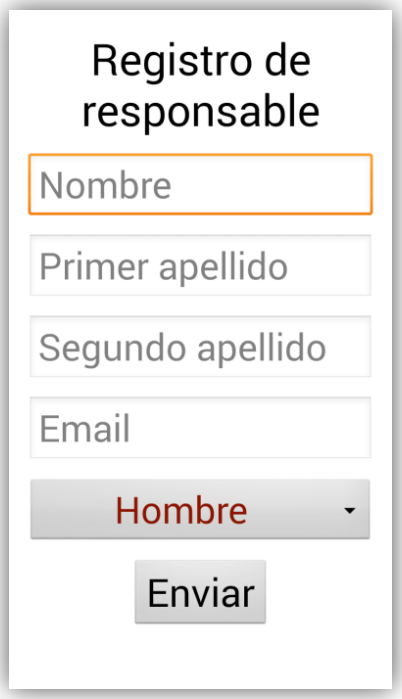


Figura 202. Pantalla de registro de responsable

B.2.4 Selección de paciente

Una vez seleccionado el responsable, como muestra el **Apartado B.2.2**, la aplicación lista los pacientes o familiares a su cargo (Figura 203), presentando una pantalla similar a la de selección de usuario en la aplicación de pacientes (**Apartado B.1.2**).

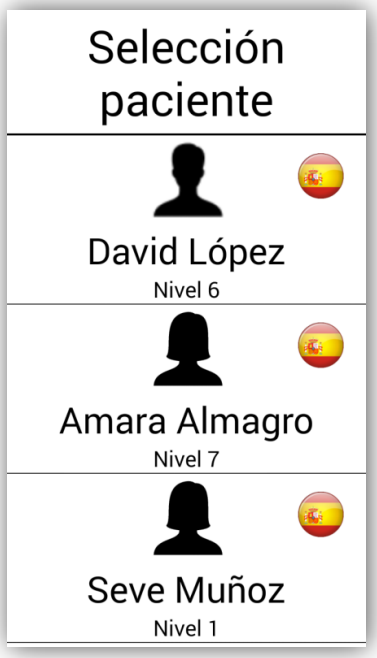


Figura 203. Selección de paciente asociado a un responsable

Si el responsable no tiene ningún paciente asociado, la aplicación lo comunica mediante un mensaje emergente y retorna a la pantalla de selección de responsable, como muestra la Figura 204.

Figura 204. Mensaje informativo para ausencia de pacientes asociados a responsable

B.2.5 Menú principal (Información paciente)

Esta pantalla, representada en la Figura 205, muestra el menú con las opciones disponibles una vez seleccionado el paciente como indica el **Apartado B.2.4**. En la parte superior se muestra una cabecera con el nombre, primer apellido y nivel actual del paciente de la terapia cognitiva. Debajo se presentan las opciones disponibles del menú:

- *Mostrar estadio*. Al pulsarlo navega a la pantalla mostrada en el **Apartado B.2.7**.
- *Estadísticas.* Enlaza con las gráficas estadísticas descritas en el **Apartado B.2.6**.
- *Otro paciente.* Permite volver a la pantalla de selección de paciente (**Apartado B.2.4**)
- *Salir*. Permite salir de la aplicación **HealthCoach Admin**.

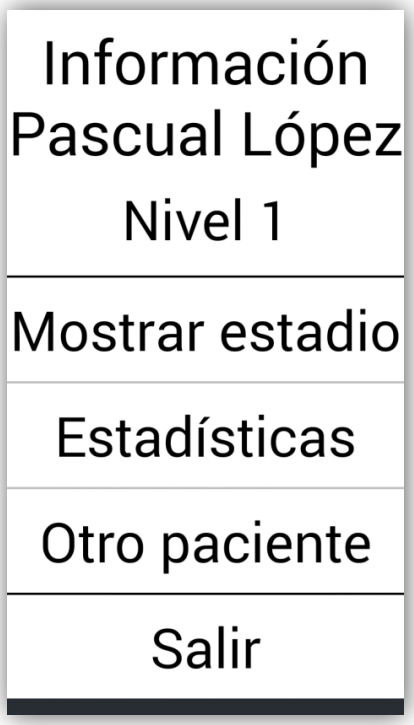


Figura 205. Pantalla de menú principal mostrada en la aplicación HealthCoach Admin

B.2.6 Estadísticas de paciente

El módulo estadístico es accesible desde el menú principal descrito en el **Apartado B.2.5**, pulsando en "Estadísticas". La funcionalidad de este módulo es idéntica a la descrita anteriormente para la aplicación de pacientes en el **Apartado B.1.6**.

B.2.7 Estadio enfermedad

Una de las características más importantes de la aplicación de responsables (*HealthCoach Admin*) es la de obtener una estimación aproximada del estadio o grado de la enfermedad de Alzheimer.

Una vez iniciado el proceso pueden darse dos situaciones:

1. El usuario del que se solicita el cálculo del grado de Alzheimer no ha realizado suficientes tarjetas que permitan realizar la estimación. De forma concreta, es necesario superar el nivel 6 de la terapia cognitiva para que el cálculo pueda llevarse a cabo. La Figura 206 muestra el mensaje mostrado por la aplicación si se da esta situación.

Figura 206. Mensaje mostrado ante la imposibilidad de estimar el estadio de Alzheimer

2. El paciente ha contestado suficientes tarjetas de la terapia para realizar el cálculo del estadio. Como muestra la Figura 207, en primera instancia se muestra un mensaje que indica el proceso de envío de correo electrónico con la información resultante del grado de la enfermedad y a continuación se muestra en pantalla de manera numérica, junto con las opciones de *Continuar* para volver a la pantalla de información de paciente (**Apartado B.2.5**) o *Resetear datos*, explicado en el **Apartado B.2.7.1**.

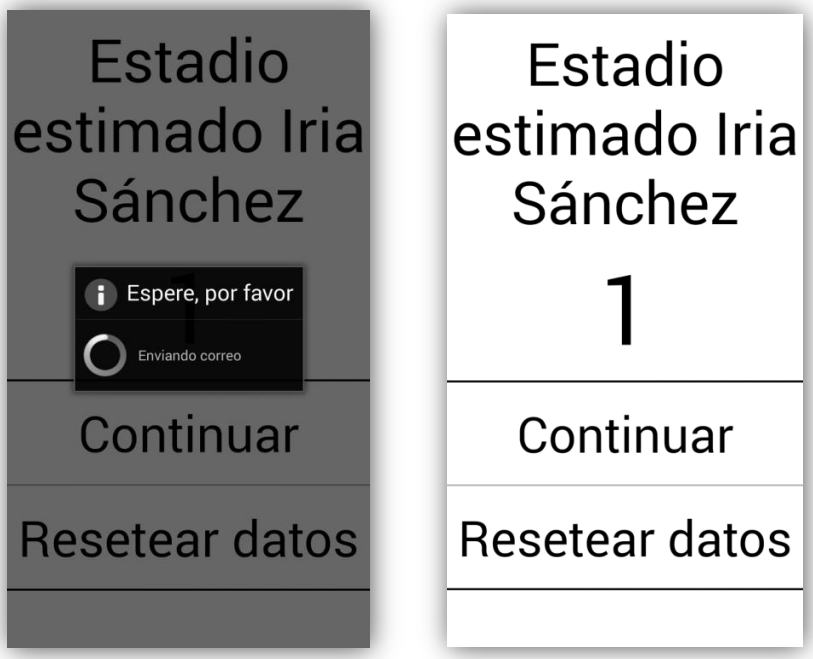


Figura 207. Proceso de cálculo del estadio de Alzheimer del paciente

B.2.7.1 Resetear datos de paciente

Para borrar todo progreso del paciente en la terapia cognitiva, que no los datos o cuenta del paciente en sí, se habilita la opción de *Resetear datos*. Una vez pulsada se mostrará un mensaje de confirmación como el mostrado en la Figura 208.

Figura 208. Mensaje de confirmación de reseteo de la evolución del paciente

B.2.8 Correo informativo

Los responsables recibirán correos electrónicos a modo de notificación cuando se produzcan eventos en la terapia cognitiva de cada uno de sus pacientes. Por ello, es fundamental que en el registro de paciente se establezca correctamente el correo electrónico del responsable, lo que posibilita la asociación entre ellos para llevar a cabo la monitorización desde la propia aplicación de responsables y recepcionar las notificaciones en forma de correo electrónico.

Son tres las circunstancias que pueden generar un correo informativo:

1. Superación de un nivel de dificultad. Al realizar todas las tarjetas de un nivel disponibles para el usuario a criterio del algoritmo de adaptabilidad. En este correo informativo se muestra la racha máxima de aciertos consecutivos, como muestra la Figura 208.

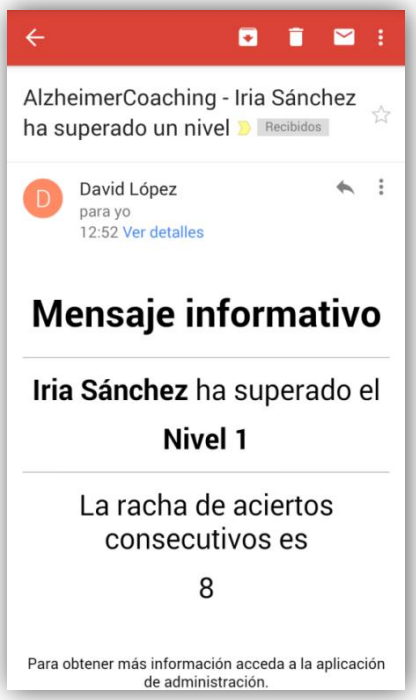


Figura 209. Correo electrónico de superación de nivel de terapia para paciente asociado

2. Resultado del algoritmo de adaptabilidad. En este correo se informa al responsable del porcentaje de aciertos por ámbito cognitivo para el nivel completado por el paciente, así como los ámbitos que estarán disponibles en el siguiente nivel y aquellos que tendrán un refuerzo especial, como puede observarse en la Figura 210.

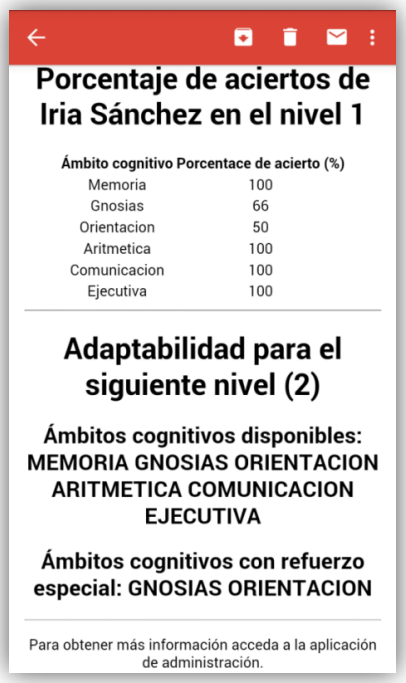


Figura 210. Correo electrónico con el resultado de la adaptabilidad en la terapia cognitiva

3. Generación del cálculo del estadio de la enfermedad. Cuando un paciente supera el nivel 6 de la terapia cognitiva, la aplicación HealhCoach calcula de forma automática el grado de enfermedad de ese paciente, comunicando el resultado a su responsable por medio de un correo electrónico como el que muestra la Figura 211.

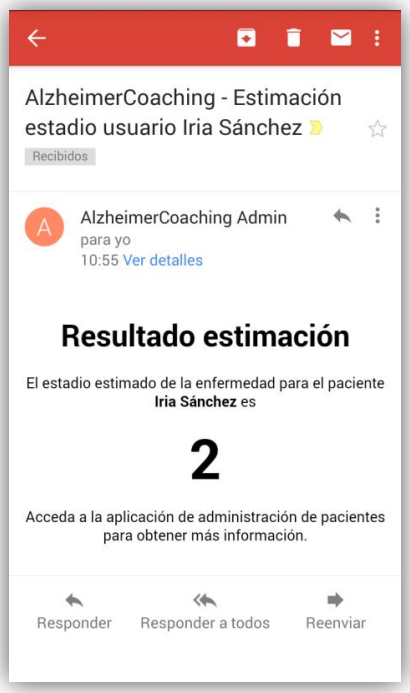


Figura 211. Correo electrónico de estimación del estadio de la enfermedad de paciente

Glosario

- **Activity:** componente utilizado en las aplicaciones Android para representar las pantallas que componen la interfaz de usuario.

- **ADT:** siglas de Android Development Tools**,** es un *plug-in* de Eclipse que permite el desarrollo de aplicaciones Android.

- **Apache HTTP:** servidor multiplataforma, de código abierto y gratuito destinado a servir páginas y servicios en formato HTML a través de la red.

- **ART:** siglas de *Android Runtime*, nueva máquina virtual introducida en las últimas versiones de Android sustituyendo a Dalvik. Es el entorno de ejecución utilizado a partir de Android 5.0 (Lollipop).

- **ASR**: siglas en inglés de *Automatic Speech Recognition*. Es el proceso mediante el que se reconoce la voz emitida por el usuario y devuelve un conjunto de palabras potencialmente pronunciadas, todo ello mediante el uso de algoritmos de procesado de señales de voz.

- **API:** siglas de *Application Programming Interface*, es una colección de funciones o métodos recopilados en forma de biblioteca para crear una capa de abstracción sobre la que trabaja el desarrollador. Define la forma en que un módulo software interactúa con otro.

- **APK:** siglas de *Applicacion PacKage File*, es un paquete de aplicación para la plataforma Android.

- **Asynctask:** clase Android utilizada para ejecutar tareas en segundo plano, con el propósito de evitar bloquear el hilo principal de ejecución.

- **Bytecode:** conjunto de instrucciones resultantes de la compilación de código fuente Java, compatibles con el dispositivo hardware donde se ejecuta el programa. Se guardan en archivos de extensión .class.

- **Códec:** elemento software-hardware capaz de codificar y descifrar un archivo con un flujo de datos. Muy utilizado en videoconferencias y archivos multimedia.

- **Dalvik:** entorno de ejecución de Android, derivado de una versión reducida de la máquina virtual de Java.

- **Dirección IP:** Identifica, mediante un número único, a un dispositivo dentro de una red de protocolo IP. Se forman con un conjunto de cuatro números del 0 al 255 separados por puntos.

- **DNS:** siglas de *Domain Name System*, sistema utilizado para asociar direcciones IP a nombres de dominio mediante una base de datos distribuida.

- **Driver:** programa que controla la interacción entre el sistema operativo y un periférico.

- **FTP:** siglas en inglés de *File Transfer Protocol*, es un protocolo de red sobre TCP/IP para la transferencia de archivos entre un cliente y un servidor. Normalmente utiliza los puertos 20 y 21.

- **GUI:** siglas en inglés de *Graphical User Interface*, Interfaz Gráfica de Usuario. Parte del sofware que permite la interacción con el usuario, utilizando elementos gráficos que forman un entorno visual.

- **GPL:** siglas de *General Public License*, es la licencia creada por la *Free Sofware Foundation* que define las bases de distribución, modificación y uso de software libre.

- **HTML:** del inglés *HyperText Markup Language*, estándar que define el lenguaje de marcado utilizado en el desarrollo de páginas web.

- **IDE:** siglas de *Integrated Development Environment*, hace referencia a un entorno de desarrollo de software. Normalmente compuesto por un editor de código fuente y herramientas para la compilación y depuración.

- **IoT:** siglas en inglés de *Internet of Things*, el Internet de las cosas hace referencia a la interconexión digital de objetos cotidianos a través de la red.

- **IMEI:** siglas en inglés de *International Mobile System Equipment Identity*. Es un número único de 15 o 17 cifras que identifica de forma unívoca el terminal movil en euna red GSM o UMTS.

- **JAR:** acrónimo de *Java ARchive*. Es una colección de ficheros Java agrupados en un único archivo contenedor, de formato de compresión sin perdida llamado ZIP.

- **JDK:** siglas en inglés de *Java Development Kit*, es el conjunto de herramientas de desarrollo utilizadas para la creación de programas Java. Actualmente lo suministra Oracle.

- **JSON:** siglas de *JavaScript Object Notation*, es un formato utilizado en el intercambio de información a través de una sintaxis sencilla y compatible con cualquier lenguaje de programación.

- **JVM:** siglas en inglés de *Java Virtual Machine*, la máquina virtual de Java es el entorno de ejecución de los programas Java, transformando el bytecode en código máquina específico del hardware donde se ejecuta el programa.

- **Kernel:** núcleo de un sistema operativo, encargado de proporcionar acceso al hardware y gestionar recursos del sistema mediante una capa de abstracción.

- **Layouts:** son ficheros XML que definen la estructura visual de la interfaz de usuario en Android.

- **MySQL:** sistema de gestión de bases de datos relacionales multihilo, multiusuario y estable, suministrado bajo licencia GPL.

- **Phablet:** combinación entre tablet y teléfono inteligente. En español se denomina también fableta o tabletófono.

- **PDO:** siglas del inglés *PHP Data Objects*, es una interfaz destinada a la conexión con diferentes tipos de bases de datos basada en PHP.

- **PHP:** siglas en inglés de *PHP Hypertext Preprocessor*, es un lenguaje de programación de propósito general y código abierto, ejecutado en el lado servidor y utilizado para generar contenido HTML de forma dinámica.

- **Plug-in:** componente software que se integra y amplía la funcionalidad de otro.

- **QWERTY**: disposición de teclas en un teclado físico, cuyo nombre se debe a las primeras seis letras de la fila superior de teclas.

- **RAH:** siglas de Reconocedor Automático del Habla, módulo dentro de los sistemas de diálogo que captan y procesan la voz del usuario para obtener una cadena de texto.

- **RAM:** siglas en inglés *de Random Access Memory*, es un tipo de memoria utilizada por los programas y sistemas operativos para cargar previamente las instrucciones que se ejecutarán en el procesador.

- **REST:** acrónimo de *Representational State Transfer*, arquitectura de diseño de comunicaciones entre aplicaciones Web basado en el protocolo HTTP. Su principal utilidad consiste en crear servicios Web o APIs.

- **RESTful:** termino aplicado a una API que sigue el patrón de diseño REST.

- **Smartphone:** término inglés del que surge el concepto de teléfono inteligente.

- **SDK:** siglas de *Software Development Kit*, es el kit de desarrollo de software formado por las herramientas utilizadas para desarrollar aplicaciones de una determinada plataforma o lenguaje.

- **SMS:** siglas en inglés de *Short Message Service*, es el servicio que permite el envío de mensajes cortos de texto entre teléfonos móviles.

- **SMTP:** siglas de *Simple Mail Transfer Protocol*, es un protocolo de red utilizado en el intercambio de correo electrónico entre dispositivos.

- **SSL:** siglas de *Secure Socket Layer*, es un protocolo de seguridad utilizado para establecer un canal seguro entre dos ordenadores conectados a través de Internet o de una red interna.

- **Tablet:** término inglés adoptado en español como tableta, se trata de una computadora portátil similar a un teléfono inteligente pero de mayor tamaño, entre 7 y 12 pulgadas.

- **Tethering**: término inglés de anclaje a red, proceso mediante el que un dispositivo móvil con conexión a Internet ofrece esta conexión a otros dispositivos ejerciendo de enrutador, al que se conectan mediante red Wi-Fi, Bluetooth o cable USB.

- **Touchpad:** también conocido como trackpad, es un dispositivo táctil de entrada que permite controlar el cursor a través de la interfaz gráfica de un sistema operativo.

- **Trackball**: elemento precursor del actual touchpad, consistente en una bola que detecta la rotación en dos ejes al accionarla con el pulgar, los dedos o la palma de la mano, moviendo el cursor en pantalla. Dispositivo muy utilizado en el diseño asistido por ordenador.

- **TTS**: siglas en inglés de *Text to Speech*, módulo de los sistemas de diálogo encargado de convertir cadenas de texto generadas a su entrada a lenguaje natural en el(los) lenguaje(s) disponibles(s).

- **UML:** siglas de *Unified Modeling Language*, es el lenguaje de modelado de sistemas software predominante en la actualidad. Utilizado principalmente para definir diagramas de clases y diagramas de entidad relación para bases de datos.

- **VoIP**: acrónimo de *Voice over IP*, es un sistema utilizado para enviar señal de voz analógica a través de internet mediante el protocolo IP.

- **Wi-Fi:** acrónimo de *Wireless Fidelity*, define el estándar utilizada para el envío de datos de forma inalámbrica a través de ondas de radio.

- **Web Service:** servicios que facilitan la interoperabilidad entre sistemas con independencia del lenguaje o plataforma en que se desarrollaron. Deben tener un método de interacción basado en un formato estándar como XML o JSON.

- **XML:** siglas de *Extensible Markup Language*, representa un lenguaje utilizado en el intercambio de datos dentro de la Web o entre aplicaciones, definiendo las etiquetas y atributos que forman la estructura de la información.

Bibliografía

[1] A. Urueña, E. Valdecasa, M. Ballestero, O. Urueña, R. Castro y S. Cadenas, "Las TIC en los hogares españoles", en *LA SOCIEDAD EN RED Informe Anual 2014*. Madrid: Observatorio Nacional de las Telecomunicaciones y de la Sociedad de la Información, 2015, pp. 59-63.

[2] Clarke, Modet & Cº, *Tendencias en las tecnologías móviles y sus aplicaciones*. Madrid: Fundación EOI, 2014.

[3] J. Annuzzi, L. Darcey y S. Conder, *Advanced Android Application Development,4th Edition*. Michigan: Addison-Wesley, 2014.

[4] GanttProject. Consultado agosto 2015. [En línea]. Disponible en: http://www.ganttproject.biz

[5] W. Stallings, "Introducción a los Sistemas Operativos", en *Sistemas Operativos*. Madrid: Fareso S.A, 1997, pp.47-48.

[6] M. Michán. " iOS desde cero: Alertas y Centro de notificaciones", *applesfera.com*, abril 2014 [en línea]. Disponible en http://www.applesfer a.com/tutoriales/ios-desde-cero-alertas-y-centro-de-notificaciones

[7] K.Verdnik. "How we leverage iOS push notifications", *Layer*, 19 diciembre 2013. [En línea]. Disponible en http://blog.layer.com/how-we-leverage-ios-push-notifications/

[8] Apple support. "iPad – Especificaciones técnicas", *apple.com*, 5 agosto 2013. [En línea]. Disponible en https://support.apple.com/kb/SP580?locale=es_ES&viewlocale=es_ES

[9] Apple support. "Usar Salud en tu iPhone o iPod touch con iOS 8", *apple.com*, marzo 2014. [En línea]. Disponible en: https://support.apple.com/es-es/HT203037

[10] Windows Phone. "Novedades en Windows Phone 8.1", *windowsphone.com*, 2014. [En línea]. Disponible en: https://www.windowsphone.com/es-ES/How-to/wp8/basics/whats-new-in-windows-phone

[11] Microsoft. "Apps for Windows Phone", *microsoft.com*, 2014. [En línea]. Disponible en: http://www.windowsphone.com/es-es/store

[12] Microsoft. "Conoce a Cortana", *microsoft.com*, 2014. [En línea]. Disponible en: https://www.windowsphone.com/es-ES/How-to/wp8/cortana/meet-cortana

[13] Mozilla Foundation. "Firefox Marketplace", *firefox.com*, 2014. [En línea]. Disponible en: https://marketplace.firefox.com/

[14] Ubuntu developer. "QML Apps for SDK 15.04", *ubuntu.com*, 2014. [En línea]. Disponible en: https://developer.ubuntu.com/api/qml/current/

[15] Ubuntu developer. "HTML5 Apps for SDK 15.04", *ubuntu.com*, 2014. [En línea]. Disponible en: https://developer.ubuntu.com/api/html5/current/

[16] BQ. "Aquaris E4.5 Ubuntu edition", *bq.com*, 2014. [En línea]. Disponible en: http://www.bq.com/es/ubuntu.html

[17] International Data Corporation. "Android and iOS Squeeze the Competition, Swelling to 96.3% of the Smartphone Operating System Market for Both 4Q14 and CY14, According to IDC", *idc.com*, febrero 2015. [En línea]. Disponible en: http://www.idc.com/getdoc.jsp?containerId=prUS25450615

[18] International Data Corporation. "Despite a Strong 2013, Worldwide Smartphone Growth Expected to Slow to Single Digits by 2017, According to IDC", *idc.com*, marzo 2015. [En línea]. Disponible en: http://www.idc.com/getdoc.jsp?containerId=prUS24701614

[19] I. Nass. "The Mobile App Market is changing in 2015, try and keep up", dazeinfo.com, 19 marzo 2015. [En línea]. Disponible en: http://dazeinfo.com/2015/03/19/the-mobile-app-market-is-changing-in-2015-try-and-keep-up/

[20] Ariel. "Most app developers stick with one store", *appfigures.com*, 23 julio 2014. [En línea]. Disponible en: http://blog.appfigures.com/most-app-developers-stick-with-one-store/

[21] J. Llisterri, "Introducción a los sistemas de diálogo", en Llisterri, Joaquim et al., *Los sistemas de diálogo*. Bellaterra – Soria: Universitat Autònoma de Barcelona – Fundación Duques de Soria, 2006, pp. 11-21.

[22] R. López-Cózar Delgado, "Sistemas de diálogo hablado y multimodal", *ugr.es*, 2006. [En línea]. Disponible en: http://www.ugr.es/~rlopezc/sistemas_dialogo.htm

[23] D. Griol Barres, *Desarrollo y Evaluación de Diferentes Metodologías para la Gestión Automática del Diálogo*. Tesis doctoral. Valencia: Universidad Politécnica de Valencia, 2007 [en línea]. Disponible en: http://users.dsic.upv.es/~dgriol/papers/TesisDgriol.pdf

[24] R. López-Cózar y M. Gea. "Sistema de Diálogo Ubicuo para Entornos Educativos", *aipo.es*, 2009. [En línea]. Disponible en: http://aipo.es/articulos/3/15.pdf

[25] M.I. Torres, "El reconocimiento del habla", en Llisterri, Joaquim et al., *Los sistemas de diálogo*. Bellaterra – Soria: Universitat Autònoma de Barcelona – Fundación Duques de Soria, 2006, pp. 11-21.

[26] E. Sosa, "Procesamiento del lenguaje natural: revisión del estado actual, bases teóricas y aplicaciones (Parte I)", *El profesional de la información, Vol.6, nº1-2*, enero 1997 [en línea]. Disponible en: http://www.elprofesionaldelainformacion.com/contenidos/1997/enero/procesamiento_del_lenguaje_natural_revisin_del_estado_actual_bases_tericas_y_aplicaciones_parte_i.html

[27] VoiceWeb. "Voice Ticketing "Would you like to buy a ticket today?"", *voiceweb.eu*, 2011. [En línea]. Disponible en: http://www.voiceweb.eu/content/serviceFiles/service_25/voiceTicketing.pdf

[28] Natural vox. "La Experiencia Natural en tus Comunicaciones", *naturalvox.com*, 2011. [En línea]. Disponible en: http://www.naturalvox.com/

[29] F. J. Hernando Pericas, J. Padrell Sendra y H. Rodríguez Hontoria, "). Sistema de información meteorológica automática por teléfono ATTEMPS", *Procesamiento del lenguaje natural*, nº29, pp. 311-312, septiembre 2002 [en línea]. Disponible en: http://hdl.handle.net/10045/1765

[30] C. Vaquero, O. Saz, E. Lleida, J. M. Marcos y C. Canalís, "VOCALIZA: AN APPLICATION FOR COMPUTER-AIDED SPEECH THERAPY IN SPANISH LANGUAGE", en *IV Jornadas en Tecnología del Habla*, pp 321-326, noviembre 2006 [en línea]. Disponible en: http://lorien.die.upm.es/~lapiz/rtth/JORNADAS/IV/finals/4jth_111.pdf

[31] J.Gabriel Amores, G. Pérez y P. Manchón. "MIMUS: a multimodal and multilingual dialogue system for the home domain". Presentado a: *Proceedings of the ACL 2007 Demo and Poster Sessions*, Praga, junio 2007 [en línea]. Disponible en: http://www.aclweb.org/anthology/P07-2001

[32] E. Turner. "How does Siri work?", *quora.com*, 10 octubre 2011. [En línea]. Disponible en: https://www.quora.com/How-does-Siri-work

[33] J. Gustafson, L. Bell, J. Beskow, J. Boye, R. Carlson, J. Edlund, B. Granström, D. House, David y M. Wirén. (2006). *AdApt – a multimodal conversational dialogue system in an apartment domain*. Suecia: Centre for Speech Technology, Royal Institute of Technology, 2006 [en línea]. Disponible en: http://www.diva-portal.org/smash/get/diva2:642843/FULLTEXT01.pdf

[34] Andalinux. "Escribir en móvil con Android sin tocar el teclado", *Informático de Guardia*, octubre 2011. [En línea]. Disponible en: https://andalinux.wordpress.com/2011/10/10/escribir-en-movil-con-android-sin-tocar-el-teclado/

[35] K. Puerto. "Google Maps Navigation en España: un navegador GPS y búsquedas por voz en tu teléfono Android", *Xataka móvil*, junio 2010. [En línea]. Disponible en: http://www.xatakamovil.com/aplicaciones/google-maps-navigation-en-espana-un-navegador-gps-y-busquedas-por-voz-en-tu-telefono-android

[36] Google. "Ayuda de Nexus. Introducir texto por Voz", *google.com*, 2014. [En línea]. Disponible en: https://support.google.com/nexus/answer/2781851?hl=es

[37] H. Barra y D. Burke. "Just speak it: introducing Voice Actions for Android", Google Mobile Blog, 12 agosto 2010. [En línea]. Disponible en: http://googlemobile.blogspot.com.es/2010/08/just-speak-it-introducing-voice-actions.html

[38] A. Pérez Figueras. "Las Acciones de Voz para Android en español", *Blog Oficial de Google España*, 15 septiembre 2011. [En línea]. Disponible en: http://googleespana.blogspot.com.es/2011/09/las-acciones-de-voz-para-android-en.html

[39] X. Santamaria. "Google Now: Historia, evolución y uso del asistente personal de Android", *Andro4all*, diciembre 2014. [En línea]. Disponible en: http://andro4all.com/2014/12/google-now-historia-evolucion-uso

[40] J. Wilkiewicz. "The fastest route between voice search and your app", Android Developers Blog, 29 octubre 2014. [En línea]. Disponible en: http://android-developers.blogspot.com.es/2014/10/the-fastest-route-between-voice-search.html?=1

[41] G. González. ""Ok Google" ya habla español", *hipertextual.com*, 3 julio 2014. [En línea]. Disponible en: http://hipertextual.com/archivo/imagen-del-dia/ok-google-habla-espanol/

[42] M. Ramírez. "Todos los comandos de voz de Google Now para usar con "Ok, Google"", *Androidsis*, 3 julio 2014. [En línea]. Disponible en: http://www.androidsis.com/todos-los-comandos-de-voz-de-google-now-para-usar-con-ok-google/

[43] Google. "Package android.speech.tts", *Android Developers*, 21 mayo 2015. [En línea]. Disponible en: http://developer.android.com/reference/android/speech/tts/package-summary.html

[44] Google. "An Introduction to Text-To-Speech in Android", *Android Developers*, Septiembre 2009. Disponible en: http://android-developers.blogspot.com.es/2009/09/introduction-to-text-to-speech-in.html

[45] V. Isidro Carretero, C. Pérez Muñano, V. Sánchez-Valladares Jaramillo y A. Balbás Repila, *Guía práctica para familiares de enfermos de Alzheimer*. Centro Alzheimer Fundación Reina Sofía-Clece Servicios Sociales, 2011. [E-book] Disponible en: http://www.fundacionreinasofia.es/Lists/Documentacion/Attachments/13/Guia%20practica%20familiares%20de%20enfermos%20de%20Alzheimer_final.pdf

[46] P. Ramos Cordero et al. *La enfermedad de Alzheimer y otras demencias*. Madrid: Dirección General de Salud Pública y Alimentación, 2007.

[47] OMS. "Demencia", *Organización Mundial de la salud*, marzo 2015. [En línea]. Disponible en: http://www.who.int/mediacentre/factsheets/fs362/es/

[48] M. T., Abellán Vidal et al. "Introducción" en *Guía de Práctica Clínica sobre la Atención Integral a las Personas con Enfermedad de Alzheimer y otras Demencias*. Plan de Calidad para el Sistema Nacional de Salud del Ministerio de Sanidad, Política Social e Igualdad. Agència d'Informació, Avaluació i Qualitat en Salut de Cataluña, 2010, pp. 61-62.

[49] B. Reisberg, S. H. Ferris, M. D. de León y T. Crook. "The global deterioration scale for assessment of primary degenerative dementia", *American Journal of Psychiatry*, vol. 139, nº 9, pp. 1136-1139, septiembre 1982.

[50] B. Reisberg. "Functional Assessment Staging Fast", *Psycopharmacol Bull*, vol. 24, pp. 653-659, 1988.

[51] B. Reisberg y S. H. Ferris. "Brief Cognitive Rating Scale (BCRS)", *Psycopharmacol Bull*, vol.24, pp. 629-636, 1988.

[52] L. Tárraga y M. Boada. *Volver a empezar. Ejercicios prácticos de estimulación cognitiva para enfermos de Alzheimer*. Barcelona: Editorial Glosa, 2007.

[53] J. Peña-Casanova. "Métodos y objetivos de intervención cognitiva en la enfermedad de Alzheimer: una visión de conjunto" en *Intervención cognitiva en la enfermedad de Alzheimer*. Barcelona: Fundación "la Caixa", 1999, pp. 9-11.

[54] E. L. Arana. "Andzheimer", *Google Play*, 2014. [En línea]. Disponible en: https://play.google.com/store/apps/details?id=net.arananet.info.andzheimer

[55] Solusoft. "Tweri Alzheimer Caregiver", *Google Play*, 2014. [En línea]. Disponible en: http://www.tweri.com/inicio.aspx

[56] T. Neila. "Alzheimer Info y Ejercicios", *Google Play*, 2014. [En línea]. Disponible en: https://play.google.com/store/apps/details?id=air.alzheimer

[57] Cimarron International. "Mid-stage Alzheimer's Cards", *Google Play*, 2014. [En línea]. Disponible en: https://play.google.com/store/apps/details?id=com.kovava.MidStageCards

[58] Supertruper. "[re]membr", *Google Play*, 2014. [En línea]. Disponible en: https://play.google.com/store/apps/details?id=com.supertruper.remembr

[59] W3C. "Extensible Markup Language (XML)", *W3C Information and knowledge domain*, 19 Mayo 2015. [En línea]. Disponible en: http://www.w3.org/XML/

[60] Java. "Conozca más sobre la tecnología Java", *java.com*, 2014. [En línea] Disponible en: https://www.java.com/es/about/

[61] W3C. "Hipertext Transfer Protocol – HTTP/1.1", *W3C Information and knowledge domain*, 2014. [En línea] Disponible en: http://www.w3.org/Protocols/rfc2616/rfc2616.html

[62] M. Rouse. "REST (representational state transfer) definition", *TechTarget* , diciembre 2014. [En línea]. Disponible en: http://searchsoa.techtarget.com/definition/REST

[63] T. Fredreich. "Learn REST: A RESTful Tutorial", *REST API Tutorial*, mayo 2012. [En línea]. Disponible en: http://www.restapitutorial.com/

[64] JSON. "Introducción a JSON", *json.org*, 2014. [En línea]. Disponible en: http://json.org/json-es.html

[65] PHP. "¿Qué es PHP?", *php.net*, 2015. [En línea]. Disponible en: http://php.net/manual/es/intro-whatis.php

[66] PDO. "Introducción", *php.net*, 2015. [En línea]. Disponible en: http://php.net/manual/es/intro.pdo.php

[67] MySQL. "The world's most popular open source database", *mysql.com*, 2014. [En línea]. Disponible en: https://www.mysql.com

[68] Apache. "Apache HTTP Server Project", *Apache HTTP Server Project*, 2014. [En línea]. Disponible en: http://httpd.apache.org/

[69] J. Lockhart, A. Smith y R. Allen. "Slim a micro framework for PHP", *slimframework.com*, 2015. [En línea] Disponible en: http://www.slimframework.com/

[70] Javamail-Android. "JavaMail port for the Android platform", *code.google.com*, septiembre 2009. [En línea]. Disponible en: https://code.google.com/p/javamail-android/

[71] AndroidPlot. "AndroidPlot", *androidplot.com*, 2015. [En línea]. Disponible en: http://androidplot.com/

[72] Java. "¿Cómo puedo empezar a desarrollar programas Java con Java Development Kit (JDK)?", *java.com*, 2014. [En línea] Disponible en: https://www.java.com/es/download/faq/develop.xml

[73] Oracle. "Java SE Development Kit 8 Downloads", *oracle.com*, 2014. [En línea]. Disponible en: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

[74] Eclipse. "Eclipse", *eclipse.org*, 2014. [En línea]. Disponible en: https://eclipse.org/downloads/

[75] Google. "SDK Manager", *Android Developers*, 2014. [En línea]. Disponible en: http://developer.android.com/tools/help/sdk-manager.html

[76] Google. "Download Android Studio and SDK Tools", *Android Developers*, 2015. [En línea]. Disponible en: https://developer.android.com/sdk/index.html#Other

[77] Google. "Android Developer Tools", *Android Developers*, 2014. [En línea]. Disponible en: http://developer.android.com/tools/help/adt.html

[78] Toad. "Toad for MySQL", *toadworl.com*, 2014. [En línea]. Disponible en: https://www.toadworld.com/products/toad-for-mysql

[79] Postman. "80+ features to make your API workflow faster", *getpostman.com*, 2014. [En línea] Disponible en: https://www.getpostman.com/features

[80] Putty. "Download Putty", *putty.org*, 2014. [En línea]. Disponible en: http://www.putty.org/

[81] Filezilla. "Filezilla Features", *filezilla-project.org*, 2014. [En línea]. Disponible en: https://filezilla-project.org/client_features.php

[82] J. Tomás Girones. *El gran libro de Android*. Barcelona: MARCOMBO, 2013.

[83] Berners-Lee, et. al. "Uniform Resource Identifiers (URI): Generic Syntax", RFC 3296, agosto 1998 [En línea]. Disponible en: https://www.ietf.org/rfc/rfc2396.txt

[84] Google. "App Manifest", *Android Developers*, 2014. [En línea]. Disponible en: http://developer.android.com/guide/topics/manifest/manifest-intro.html

[85] J. Delia. "Modelado orientado de caso de uso", *Modelo de casos de uso. Prototipos y técnicas de análisis*, 16 julio 2012. [En línea]. Disponible en: http://casodeusoytecnicasdeanalisis.blogspot.com.es/

[86] Fluid. "Bring the Fluid UI mobile prototyping workflow to your team", *fluidui.com*, 2014. [En línea]. Disponible en: https://www.fluidui.com

[87] Google. "Google Forms. Crea formularios atractivos", *docs.google.com*, 2015. [En línea]. Disponible en: https://docs.google.com/forms/create?usp=mkt_forms

[88] Google. "Google Docs. Crea documentos impactantes", *google.es*, 2015. [En línea]. Disponible en: https://www.google.es/intl/es/docs/about/

[89] R. Barranco Fragoso. "¿Qué es Big Data?", *IBM developerWorks*, 18 junio 2012. [En línea]. Disponible en: https://www.ibm.com/developerworks/ssa/local/im/que-es-big-data/

[90] Raspbian. "Welcome to Raspbian", *raspbian.org*, 2012. [En línea]. Disponible en: https://www.raspbian.org

[91] Raspbian. "Raspbian Images", *raspbian.org*, 2014. [En línea]. Disponible en: http://downloads.raspberrypi.org/raspbian_latest

[92] Win32DiskImager. “Win32 Disk Imager. A tool for writing images to USB sticks or SD/CF cards”, *sourceforge.net*, 2015. [En línea]. Disponible en: http://sourceforge.net/projects/win32diskimager/

[93] Putty. “PuTTY Download Page”, *chiark.greenend.org.uk*, 2015. [En línea]. Disponible en: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

[94] Apache. “Apache HTTP Server Tutorial: .htaccess files”, *apache.org*, 2015. [En línea]. Disponible en: http://httpd.apache.org/docs/current/howto/htaccess.html#troubleshoot

[95] No-IP. “Remote Access with Dynamic DNS. The freedom to connect to your devices from anywhere”, *noip.com*, 2015. [En línea]. Disponible en: https://www.noip.com/